SECURITY MEASURES IN WIRELESS SENSOR NETWORKS BASED ON PUBLISH/SUBSCRIBE DDS-MIDDLEWARE

BY

ABDALLAH HASAN KHALIL RASHED

A Dissertation Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

In

COMPUTER SCIENCE AND ENGINEERING

MAY 2018

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

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

Dr. Marwan Abu-Amara
(Advisor)

Dr. Ahmed Al-Mulhem
Department Chairman

Dr. Ashraf Mahmoud
(Co-Advisor)

Dr. Salam A. Zummo
Dean of Graduate Studies

Dr. Shokri Selim
(Member)

15/12/13
Date

Dr. Tarek Rahil Sheltami
(Member)

Dr. Mohammad Alshayeb
(Member)

Dedicated

to

My Beloved Parents, Wife and Brothers

ACKNOWLEDGMENTS

All praise and thanks are due to Almighty Allah, Most Gracious and Most Merciful, for his immense beneficence and blessings. He bestowed upon me health, knowledge, and patience to complete this work. May peace and blessings be upon Prophet Muhammad (PBUH), his family and his companions.

Thereafter, acknowledgment is due to the support and facilities provided by the Computer Engineering Department of King Fahd University of Petroleum & Minerals for the completion of this work.

I acknowledge, with sincere gratitude and appreciation, the valuable time and continuous guidance given to me by my dissertation advisor, Dr. Marwan Abu-Amara. I learned a lot in the academic and the personal level. I am also grateful to my co-advisor Dr. Ashraf Mahmoud and the Committee members, Dr. Shokri Selim, Dr. Tarek Sheltami and Dr. Mohammad Alshayeb for their constructive guidance and support. In, addition, I want to thank Dr. Uthman Baroudi for his collaboration and support.

My heartfelt thanks are due to my family for their prayers, guidance, and moral support throughout my academic life. My parents' advice, to strive for excellence has made all this work possible. Also, I want to thank my wife, Natalie, for the prayers and support.

Last, but not least, thanks to all my colleagues and friends who encourage me a lot in my way to the achievement of this work. You definitely provided me with the strength that I needed to choose the right direction and successfully complete my dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

DDS	:	Data Distribution Service
OMG	:	Object Management Group
QoS	:	Quality of Service
MCU	:	Microcontroller Unit
TTP	:	Trust Third Party
Pub/sub	:	Publish/Subscribe
TDDS	:	Basic TinyDDS
MAC	:	Message Authentication Code
HMAC	:	Key-Hashed MAC
NIST	:	National Institute of Standards and Technology
Msg/s	:	Message per second
PDR	:	Packet Delivery Ratio
EED	:	End-to-End Delay
IFS	:	Interference Free Scheduling

ABSTRACT

Full Name : [Abdallah Hasan Khalil Rashed]

Thesis Title : [Security measures in Wireless Sensor Networks based on Publish/Subscribe DDS-middleware]

Major Field : [Computer Science and Engineering]

Date of Degree : [May 2018]

The wireless sensor network (WSN) is an important platform that offers a variety of applications that can be used with significant interests in both academia and industry. It allows us to expand the usage of a large number of important applications in our life. It opens the door to develop applications such as object monitoring and tracking, traffic control, military applications, climate change and environmental surveillance, remote healthcare applications, infrastructure security, etc. However, the development of the sensor applications is complex due to the constraints and characteristics of the sensor networks. Therefore, there is a need for intermediate software layer to map the network applications with the sensor hardware, generally defined as a middleware. The Publish/Subscribe (pub/sub) interaction middleware has emerged as a suitable communication paradigm for large-scale distributed computing systems. That is because of the decoupling properties for the network's participants in time, space, and synchronization. This model is well suited for many distributed systems especially those which are data-centric, where the applications are interested in the produced data rather than sender identity. Data Distribution Service (DDS) is a standard for supporting real-time distributed systems based on the publish/subscribe scheme. DDS includes many categories of Quality of Service (QoS) policies easy to use. Therefore, porting DDS features and

functionalities into WSN may significantly enhance its development and performance. For WSN platforms, DDS has a light-weight middleware called TinyDDS. Developing applications in WSNs may require interacting with sensitive data. Therefore, it is important to protect the data. Securing the data can be either at the application level where the developers provide the security requirements, or at the middleware level, where the secuity requirements are implicitly supported and then, the developers can focus on the functionality of their applications. In this work, we plan to add security measures to the TinyDDS and thoroughly investigate the cost of adding the security measure on the WSN performance and constraints by means of simulation. We add a security service module to the TinyDDS. The module is transparent to the middleware, there are no modifications on the original application or the communication protocol. The results show a reasonable overhead in terms of performance metrics after adding security measures. Some variations on the structure of the secured packets are investigated to reduce the overhead.

ملخص الرسالة

الاسم الكامل: عبدالله حسن خليل راشد

عنوان الرسالة: دعم تدابير أمنية في شبكات الاستشعار اللاسلكية التي تقوم على نظم التشغيل الوسيطة القائمة على خدمة توزيع البيانات بطريقة النشر والإشتراك

التخصص: هندسة الحاسوب

تاريخ الدرجة العلمية: : شعبان 1439 هـ - (أيّار 2018 م)

تعد شبكة المستشعرات اللاسلكية (WSN) منصة مهمة توفر مجموعة متنوعة من التطبيقات التي يمكن استخدامها مع اهتمامات مهمة في كل من الأوساط الأكاديمية والصناعة. يسمح لنا بتوسيع استخدام عدد كبير من التطبيقات الهامة في حياتنا. تفتح الباب أمام تطوير التطبيقات مثل مراقبة الكائنات وتتبعها ، ومراقبة حركة المرور ، والتطبيقات العسكرية ، وتغير المناخ والمراقبة البيئية ، وتطبيقات الرعاية الصحية عن بعد ، وأمن البنية التحتية ، الخ. ومع ذلك ، فإن تطوير تطبيقات أجهزة الاستشعار معقد بسبب القيود في خصائص شبكات الاستشعار. لذلك ، هناك حاجة لطبقة وسيطة للبرامج لتعيين تطبيقات الشبكة مع أجهزة الاستشعار ، والتي يتم تعريفها بشكل عام على أنها برامج وسيطة. ظهرت البرمجيات الوسيطة للتفاعل النشر / الاشتراك (pub sub /) كنموذج اتصال مناسب لأنظمة الحوسبة الموزعة واسعة النطاق. ويرجع ذلك إلى خصائص فصل المشاركين في الشبكة في الزمان والمكان والتزامن. هذا النموذج مناسب تمامًا للعديد من الأنظمة الموزعة وخاصة تلك التي تتمحور حول البيانات ، حيث تهتم التطبيقات بالبيانات المنتجة بدلاً من هوية المرسل. تعتبر خدمة توزيع البيانات (DDS) معيارًا لدعم الأنظمة الموزعة في الوقت الفعلي استنادًا إلى نظام النشر / الاشتراك. يتضمن DDS العديد من فئات سياسات جودة الخدمة (QoS) سهلة الاستخدام. لذلك ، قد يؤدي ترقية ميزات DDS والوظائف إلى WSN إلى تحسين كبير في تطويرها وأدائها. بالنسبة إلى الأنظمة الأساسية لـ WSN ، تحتوي DDS على وسيط خفيف الوزن يسمى TinyDDS. قد يتطلب تطوير التطبيقات في WSNs التفاعل مع البيانات الحساسة. لذلك ، من المهم حماية البيانات. يمكن أن يكون تأمين البيانات إما على مستوى التطبيق حيث يوفر المطورون متطلبات الأمان ، أو على مستوى البرامج الوسيطة ، حيث يتم دعم متطلبات الأمان ضمنيًا ومن ثم ، يمكن للمطورين التركيز على وظائف تطبيقاتهم. في هذا العمل ، نخطط لإضافة إجراءات أمان إلى TinyDDS والتحقيق بدقة في تكلفة إضافة مقياس الأمان على أداء WSN والقيود بواسطة المحاكاة. نقوم بإضافة وحدة خدمة أمان إلى TinyDDS. الوحدة النمطية شفافة للوسيطة ، لا توجد تعديلات على التطبيق الأصلي أو بروتوكول الاتصال. تظهر النتائج مقدارًا معقولًا من حيث مقاييس الأداء بعد إضافة إجراءات الأمان. تم التحقق من بعض الاختلافات في بنية الحزم المضمونة لتقليل تكلفة الإضافات على النظام الأصلي.

CHAPTER 1

INTRODUCTION

A traditional wireless sensor network (WSN) consists of a group of multiple nodes that are usually inexpensive. In the typical single sink base-station WSN applications that is seen in Figure 1, data-flow moves from the sensor nodes to the monitoring application passing eventually by the sink node. By using a one-to-many communication scheme, sensors collect data and then send them to the sink node [1]. Therefore, sensors do not take any action on the collected data. Alternatively, in wireless sensor/actuator networks (WSAN), the concept of in-network decision making is developed. The network can take action on-site without requiring external applications' action [2] as seen in Figure 2. For the purpose of brevity, we will refer to both WSN and WSAN simply as WSN in the rest of our work.

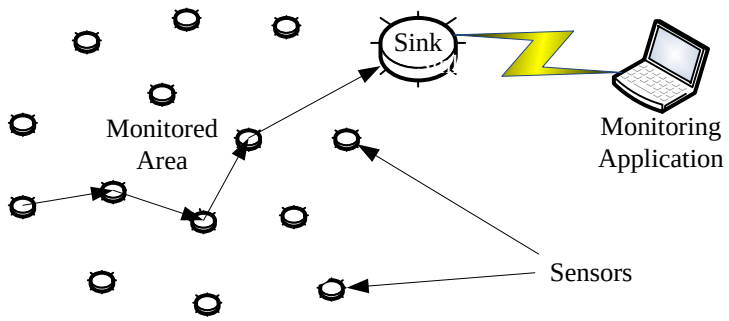


Figure 1: Traditional WSN architecture [3]

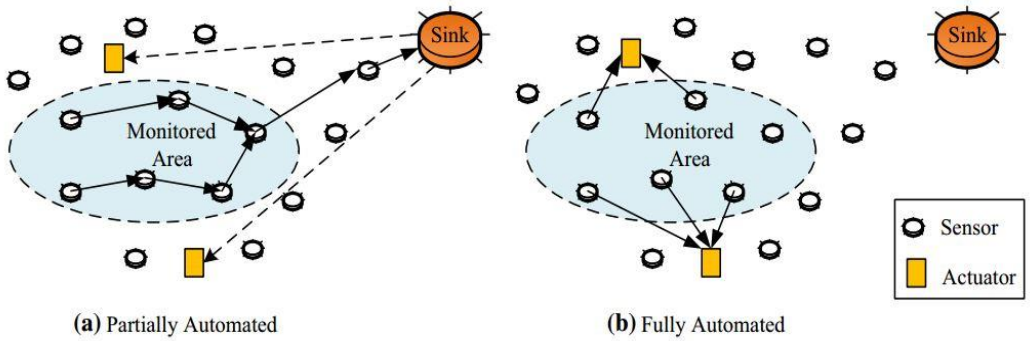


Figure 2: WSAN architecture with interaction types [3]

The nodes of a WSN are usually distributed in undesirable or harsh environmental conditions. That makes them vulnerable to in place attacks, such as tampering and jamming attacks. Moreover, there are many applications where WSNs interact with sensitive data, and the sensor nodes may be located in a challenging environment, or are hardly reachable. Therefore, it is essential to handle the security issues of confidentiality, integrity, and availability (CIA) while designing the WSN applications. However, wireless sensors have constraints with respect to the computational power and energy sources. Therefore, utilizing the conventional security methods in WSN is hard to apply efficiently [4]. The threats in WSN directly correspond to the WSN application. Therefore, a WSN application can be secured either by implementing the security aspects in the WSN application or by securing the underlying WSN the application is running on top of.

As far as the application of the security aspects for the purpose of securing the WSN application is concerned, the development of WSN applications is complex due to the constraints and the characteristics of the WSN. Hence, a middleware acts as an intermediate software layer between the WSN applications and the underlying WSN hardware. The middleware is used mainly to hide the complexity of underlying layers and to help the programmers to develop WSN applications easily. The publish/subscribe (pub/sub)

interaction middleware makes it an appropriate solution for real-time and large-scale distributed computing systems such as WSN [5-7]. Subsequently, it is desirable to modify the most suitable middleware for WSNs to account for security while considering the WSN limitations of the computational power and energy sources.

1.1 Problem Statement

As stated earlier, there are challenges with respect to securing WSNs due to their resource constraints. Therefore, it is infeasible to merely apply the security measures that are used for traditional networks directly. Many proposed security WSN schemes in the literature focus on achieving secure routing protocols, and/or secure data aggregation protocols. In this work, we consider selecting an appropriate pub/sub middleware such as TinyDDS [8] for WSNs for the purpose of enhancing it to account for the confidentiality, integrity, and availability (CIA) security requirements. Furthermore, we characterize the effect of these enhancements on the WSN performance and constraints.

1.2 Research contribution

This subsection lists the main contributions in our research.

1. Summarize of the existing middlewares in the literature which account for information security in their implementation. The summary is based on the open

source availability, operating system, sensor devices platform, information security support, and the cryptographic algorithms applied.

2. Summarize some of the main block cipher algorithms in the literature and their impact on the in WSNs. We considered the recommendations regarding the conventional and lightweight block cipher algorithms, and select suitable algorithms based on the energy consumption and memory occupancy.
3. Support information security for TinyDDS, by adding a security service module to the TinyDDS. The module is transparent to the middleware.
4. The security enhancements in the middleware aim to provide data confidentiality, integrity, and message authentication. The simulation results for confidentiality algorithms are, AES, Present, and Prince. The algorithms used for integrity are, SHA1 and MMH.
5. Regarding authentication, we implemented our devised protocol to ensure sensor nodes mutual authentication and session keys exchange. The protocol is inspired by the Kerberos protocol. We showed how the protocol is immune to some of the common attacks such as, Replay attack, Man-in-the middle attack, and Impersonation attack.

1.3 DDS publish/subscribe basic model and components

The interaction scheme of the publish/subscribe communication is well-fitted for large-scale distributed systems due to its flexibility and scalability aspects. The two main entities in the architecture of this system are the Publisher and the Subscriber. The subscription

process generally takes place when a subscriber node expresses its interest in an event and request to be notified accordingly. This interest is registered or subscribed to with publisher, and when such an event exists, the publisher notifies the subscriber. The core of the publish/subscribe scheme is the notification service which acts as a mediator between the publishers and the subscribers. It gives the flexibility and scalability aspects of this communication model through the decoupling properties in space, time and synchronization [9]. In space decoupling, the publishers and the subscribers indirectly interact with the events through the notification service; they do not need to know each other or the location of each other. In time decoupling, it is not necessary that both the publisher and the subscriber are active at the same time. For instance, a subscriber can express its interest in an event, that is not published yet, and a publisher can publish an event without being subscribed by any subscriber yet. Lastly, in synchronization decoupling, the publisher and the subscribers are not blocked during their publishing or subscribing to events.

Figure 3 depicts the standard DDS architecture. The root instance in the middleware is the *DomainParticipant*, which is a class initiated once. It maintains a list of all other instances in the middleware. In the application at the top of the middleware, an instance of *Publisher* class is initiated from the *DomainParticipant*. A *Topic* class has a value that represents an identifier of the content of an event. The application can declare as many instances of topics as the *Publisher* is interested in. On the Subscriber side, an instance of *Subscriber* class is initiated at the lowest level to monitor and capture the corresponding topic from the network traffic. The topic becomes associated with class *SubscriberListener* and class *DataReader* whenever there is a match on the content of the topic. Then, the instance

SubscriberListener informs the application about the matched event, where the data can be read from the instance *DataReader*. On the Publisher side, the application declares instances of class *DataWriter*, which is responsible for publishing the events associated with every topic declared. The application specifies the events to publish, through the use of *DataWriter*. The class *Publisher is* responsible for publishing the *DataWriter* instances to the lower-level in the network.

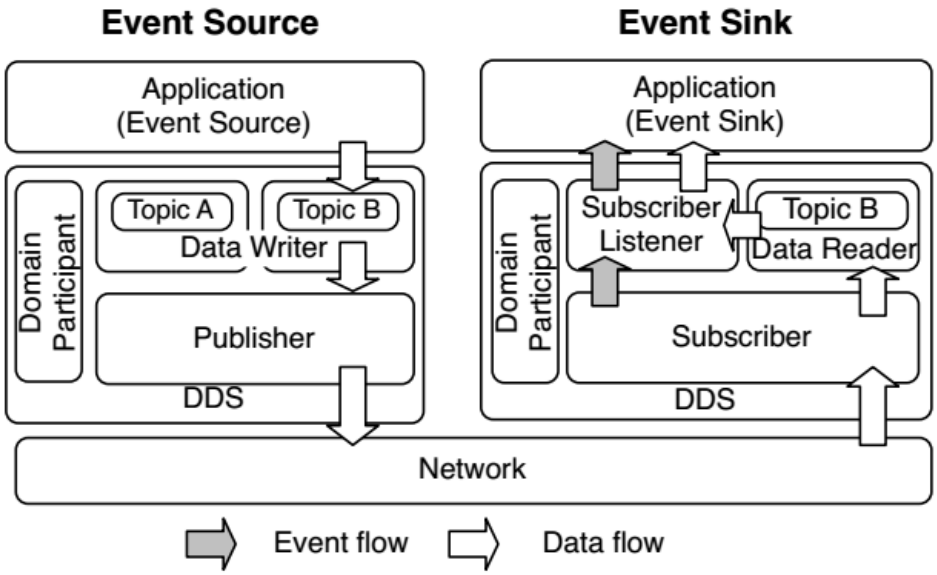


Figure 3: Standard DDS architecture [8].

CHAPTER 2

LITERATURE REVIEW

Chapter 2 reviews the existing approaches and methods in the literature regarding middleware and the security measurements for wireless sensors network. The literature review is divided into three subsections. The first subsection reviews the previous and the current classifications and approaches of WSN middlewares found in the literature. Whereas the second subsection focuses on the publish/subscribe communication approach for middlewares. It explores the importance and the benefits of this approach in developing WSN applications. The third subsection provides a description of WSN middlewares found in the literature that support secure communication specifically through cryptography.

2.1 Middleware Classifications and Approaches

The main purpose of a middleware is to hide the complexity of the layers of the underlying hardware and to conceal the heterogeneity that can exist in a network as seen in Figure 4. The authors in [10] provide a comprehensive survey of WSN middleware proposals and approaches. They built a reference model to classify and compare several WSN middlewares. In addition, they listed many ways on how the middleware can help programmers in applications' development. Such as, providing the necessary system abstractions, so that the developer focuses on the functionality of the application without worrying about the details of the lower level communication implementations. Also, the

developer can benefit from the reusable code and data services, such as data filtering, so that they can be applied easily into the applications. Another middleware classification is based on the several factors to consider in designing a middleware for WSNs including operating system support, routing protocols defined, and sensor nodes' specifications [11-13]

Moreover, several research papers describe the different middlewares that are designed for WSNs. For example, the study in [12] explores the approaches used in middlewares specific for WSNs. Also, the study presents the challenges in designing WSN middleware, such as support for heterogeneity, mobility, and scalability. Likewise, the authors in [13] present an extensive survey on service-oriented middleware for WSNs. The goal of the service-oriented middleware is to provide standardized models and protocols for the WSNs regardless of the implementation of the underlying infrastructure. As a result, services become easily available and accessible to the WSNs.

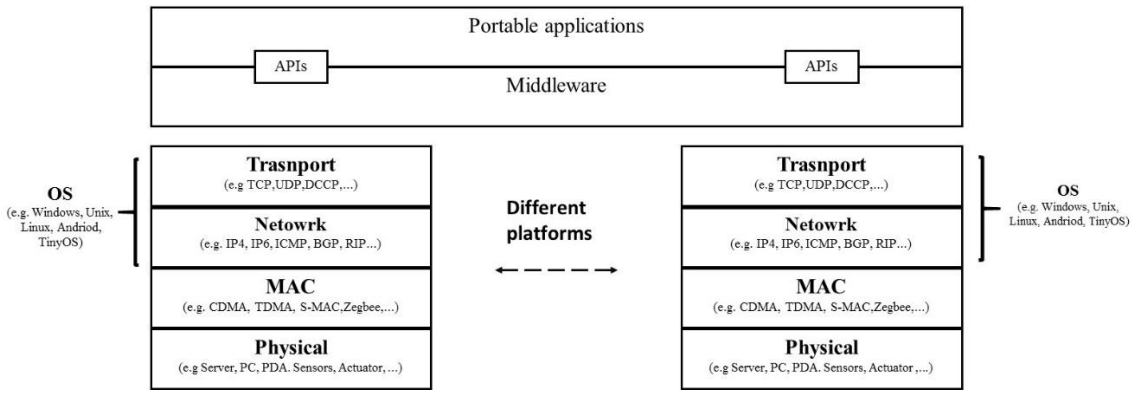


Figure 4: Middleware layer hiding heterogeneity of under layers [3]

A comparative study of different WSN middleware approaches is presented in [14]. The main classifications used are the following:

- **Database:** The WSN is represented in the middleware as a virtual database. The advantage of such approach for the middleware is the simplicity of implementation. Accordingly, the WSN application commands simply become database queries and results in easier communications. However, such an approach suffers when supporting real-time applications.
- **Event-based:** The middleware uses publish/subscribe communication scheme to simplify exchanging messages between the source and destination nodes. The main feature of this approach is supporting asynchronous communications that facilitate a loose coupling between the nodes in the network.
- **Application driven:** The applications in the middleware are in control of the network management through supporting quality of service (QoS) parameters. The problem with this approach is the lack of heterogeneity in platforms support.
- **Modular programming:** The application in this approach is divided into smaller modules that can be separately updated. It reflects positively on the energy consumption. However, the approach does not support heterogeneous platforms.
- **Virtual machine:** The middleware runs as a virtual machine on the WSN nodes. The advantage of this approach is the ability to develop independent small size modules of the WSN application that can be executed on heterogeneous platforms using interpreters.
- **Tuple space:** This middleware approach applies the shared memory paradigm that is used in distributed computing systems to the WSN nodes.

Table 1 expands on the work conducted in [14] by providing additional examples of middleware.

Table 1: Middleware approaches

Middleware Approach	Example
Database	SINA[15], Cougar [16], DsWare [17], TinyDB [18]
Event Based Message Oriented	Mires [19], TinyCubus [20], Runes [21]
Application Driven	MiLAN [19], AutoSec [22]
Modular	Impala [23]
Virtual Machine	Mate [24], Magnet [25], Agilla [26]
Tuple Space	Teeny Lime [27], Tiny Lime [28], TS-Mid [29]

2.2 Publish/Subscribe Middlewares

A pub/sub middleware is considered an event-based middleware as explained earlier subsection 2.1. The data-centricity and decoupling properties of the pub/sub interaction scheme make it an appropriate solution for real-time and large-scale distributed computing systems [5]. It provides efficient usage of WSN limited resources and matches well with their nature. The survey paper [3] studied and classified several pub/sub WSN middlewares based on the type of message exchange and their interaction model. The interaction models are channel-based, topic-based, content-based, and type-based, and are further explained in [3].

The pub/sub interaction scheme has several advantages over other communication models. For example, it is more efficient than client-server in latency and bandwidth utilization. The pub/sub model reduces the overhead required because it does not need a request for each new data generated. This reduction of outgoing request messages improves the pub/sub latency and saves network resources. Furthermore, it is capable of supporting one-to-many connectivity with redundant publishers and subscribers. Therefore, it makes suitable for reconfigurable dynamic applications, such as in WSNs, in a robust manner.

More importantly, it maps well to connectionless protocols. For instance, it can take advantage of multicast technology to efficiently send data to multiple users [30, 31].

TinyDDS [8] is an example of the event based pub/sub middleware approach. It is compliant with the standard specifications of Object Management Group (OMG)'s Data Distribution Service (DDS) for WSNs. It is a lightweight version of the pub/sub DDS standard that offers flexibility and interoperability to the application in WSNs regardless of the programming language used in the application, and protocols supported in WSNs. Moreover, TinyDDS allows the developers to be able to have over fine-grain control on the WSN applications at the application-level and middleware-level. It provides flexibility in customizing and decoupling of non-functional properties such as event filtering, data aggregation, data storage, caching policies and routing through a pluggable framework.

The non-functional properties at the application and middleware level are forming a library, called the TinyDDS library. Providing non-functional properties such as data aggregation and event detection that are frequently used in WSN applications, results in accelerating the application development. Hence, the developers can focus on the functionality more such as interpreting the process data and event. The maintainability will be improved due to the utilization of the non-functional properties that reduce the complexity of the applications. For the middleware-level non-functional properties, developers can change the behavior of the middleware to meet the constraints desired, such as adjusting the QoS of the middleware's component or changing the routing protocol. The library of TinyDDS is portable which can be used by multiple TinyDDS applications. Utilizing the non-functional properties will result in more reusable, maintainable and better performance in developed applications. The features of TinyDDS middleware and the advantages of the

pub/sub interaction scheme make TinyDDS the most suitable WSN middleware to add the security measures to. By adding security to the non-functional properties will lead in developing applications that are secure.

There are several pub/sub middlewares for WSNs other than TinyDDS. For example, Mires [19] TinyCubus [20], and Runes [21] are similar to TinyDDS in that they implement pub/sub communication on TinyOS [32] and allow flexibility in customizing and decoupling application-level non-functional properties. However, unlike TinyDDS, they do not consider the middleware-level non-functional properties and interoperability across the boundary of WSNs and access networks. On the other hand, regarding the support of QoS, and nodes mobility. The Mires middleware does not support QoS such as, *Deadline*, *Priority*, and *Reliability*. Also, it does not support node mobility. The TinyCubus middleware support *Reliability* and node mobility. Whereas, the Runes middleware support only *Reliability* QoS through external components. Also, it supports nodes mobility. The TinyDDS middleware supports *Deadline, Priority,* and *Reliability*, but does not support the nodes mobility.

SMC [33] is another pub/sub middleware specific for body-area sensor networks. It is assumed that it will operate on a Java VM atop a powerful Linux node. Hence, SMC does not account for memory footprint and power. Whereas, TinyDDS assumes resource-limited nodes as its target platform. In addition, SMC does not provide flexibility in middleware-level non-functional properties, and interoperability in programming language and protocol as TinyDDS does. DsWare [17] is yet another pub/sub middleware that provides flexibility only at middleware-level non-functional properties but not at the application-level nor providing interoperability in programming language and protocol as TinyDDS does.

2.3 Middlewares with Security Support

Several middleware platforms support security in their services through cryptography. SM-Sens [34] is a middleware that provides security on data through message authentication and using symmetric and asymmetric cryptography. Also, it takes advantage of the hierarchical routing to distribute cryptographic keys. Moreover, it uses intrusion detection to exclude compromised nodes in the network. However, it does not support programming language interoperability across platforms. STaR [35] is a transparent configurable component middleware for securing communication in WSNs in which the confidentiality, integrity, and authentication are provided through encryption. It is considered a transparent middleware because there is no need to change the communication protocol and the application itself. It supports multiple security policies at the same time, called traffic flow policy. The STaR middleware can be classified among the **Modular** middlewares category discussed in subsection 2.1. It offers the ability to exclude some of its modules in order to save memory. SpartanRPC middleware [36] provides a link-layer remote procedure call (RPC) scheme by extending nesC [37] programming language, which is protected by language-level policy specification. Moreover, it incorporates the security architecture in a heterogeneous environment. Also, the information security supported in the middleware account for data confidentiality using the AES algorithm. SMEPP Light middleware [38] is based on a subscribe/event mechanism which provides group management, group-level security policies, methods for query injection and data collection. It is similar to TinyDDS in that it implements pub/sub communication on TinyOS. However, it is not an open source middleware. The authors in [39] accounts for the availability security requirement through intrusion detection systems in addition to cryptography methods for protecting data. Other

middleware platforms provide security using defense strategies against specific attacks. For instance, Di-Sec [40] provides a defense framework through training phase that makes the nodes able to the behavior to adapt to attacks. The variety of security policy in the middleware allows to setup multiple security policies at the same time, and it can be changed at the runtime for the packets.

In [41], the authors propose Component and Policy Infrastructure (CaPI) middleware that includes security. The tool is mostly focused on the opportunity to include security during application development. It offers two first-class programming abstractions, components, and policies. Components implement coarse-grained and reusable units of software functionality that are used to realize functional application concerns. Resource constraints dictate that these components must allow full access to low-level features provided by heterogeneous OS and hardware platforms. In contrast, policies are a more lightweight abstraction, implemented in an efficient, expressive, platform-independent language that is designed to support customization and management of behavioral concerns at runtime.

ESCAPE [42] is a policy-based WSN middleware implemented on TinyOS. Applications in ESCAPE are implemented using nesC-components and adopt a similar pub/sub interaction model as in CaPI. However, ESCAPE differs from CaPI in a significant number of elements. First, the set of components and policies on a node in ESCAPE is not dynamically adaptable. Policies and application components cannot be dynamically installed, removed, nor recomposed. At compile time, ESCAPE policies are statically integrated together with the application code. As a result, policies can only be activated and deactivated at runtime. Second, ESCAPE policies are used to specify all application behavior, including the definition of distributed interactions. For instance, policies are used

to schedule timers to sample data, handle time-outs, filter resulting measurements, and publish events over the network.

In [43] a security architecture that integrates, enriches, and extends a component-based middleware layer with abstractions and mechanisms for secure reconfiguration and secure communication. The security policies are processed by a module external to the middleware.

Table 2 provides a summary of the existing middlewares in the literature which account for information security in their implementation. The summary includes the open source availability, operating system, sensor devices platform, information security support, and the secure algorithms used.

Table 2: Summary of some of secure middleware in the literature.

Middleware	Open source	Operating system	Sensor Platform	Security measure support	Security Algorithms
SM-Sens [34]	Yes	TinyOS	Berkeley/Crossbow MicaZ [44]	Confidentiality, Integrity, and Authenticity.	HMAC [45], ECC [46], RC5[47]
StaR [35]	Yes	TinyOS	Tmote Sky motes[48], CC2420 [49]	Confidentiality, Integrity and Authenticity.	Skipjack encryption module [50], ShA-1 module integrity hashing [51]
SpartanRPC [36]	Yes	TinyOS	Tmote Sky motes[48], CC2420 [49]	Confidentiality and Authenticity	AES [52], Sprocket [53]
SMEPP Light [38]	No	TinyOS	Berkeley/Crossbow MicaZ [44]	Confidentiality and Integrity	
CaPI [41]	No	Contiki 2.5 [54]	AVR-Raven [55]	Confidentiality	XTEA cipher [56]
ESCAPE [42]	Yes	TinyOS	Tmote Sky motes[48]	Confidentiality	Skipjack encryption module [50]
Dini and Savino [43]	No	Contiki 2.5 [54]	Tmote Sky motes[48]	Confidentiality and Integrity	ShA-1 [51] , RC5[47]

CHAPTER 3

METHODOLOGY AND PROPOSED APPROACH

In this chapter, the methodology and the tentative plan of our work is presented along with the phases and their tasks for our work progress. Discussion on the existing cryptographic algorithms and compare their performance on different sensor platforms to decide which algorithm to select for our implementation of the security service module. To estimate the effect of implementing our security service, in the next section, we define the performance metrics used in the evaluation of our implementation for modified TinyDDS and lastly, we present the simulation tools used to run the experiments.

3.1 Methodology

PHASE 1: (objective 1)

In this phase, we will conduct an extensive literature review and survey. It is divided into three tasks as follows:

> **Task 1:** Survey the previous work in security targeting the confidentiality, integrity and availability requirements in WSNs. The purpose is to identify obstacles, major needs, types of attacks and current defenses in WSNs.
>
> **Take 2:** Review the current pub/sub interaction schemes for WSNs to point out the schemes' main concepts, components, architectures, and variants.

Task 3: Survey the previous work related to secure middleware for WSNs, and summarize the findings in a comparison study.

Approach: Exhaustive search will be undertaken to find the state-of-the-art of the relevant high-quality research.

PHASE 2: (objective 2)

The modified-TinyDDS middleware for WSNs will be proposed in this phase.

Task 1: Modified-TinyDDS development: We are going to modify the TinyDDS to add security components for the purpose of achieving information security. We are mainly targeting the confidentiality and integrity security requirements of the data through means of encryption/decryption methods. In addition, we consider the aspect of availability as a security requirement for WSNs. In achieving these security requirements, we take into consideration the resource constraints in wireless sensor nodes, especially the memory size and power consumption.

Approach: Identify improvements related to WSN security requirements to be added to TinyDDS guided by the intensive survey from the previous phase.

PHASE 3: (objective 3)

The impact of the security modifications added to the TinyDDS on the WSN performance, and QoS support will be thoroughly investigated in this phase.

Task 1: Preparing the simulation environment, and building the testing scenarios for the modified-TinyDDS.

Approach: TOSSIM simulator [57] will be used to simulate the modified-TinyDDS (pub/sub technique).

Task 2: Evaluate the performance of the modified-TinyDDS, and collect and analyze the results.

Approach: A performance evaluation comparative study will be conducted between the modified-TinyDDS and the basic TinyDDS. The performance metrics pertaining to memory footprint, energy consumption, and communication overhead will be considered. The tradeoffs between security components added will be investigated through the analysis of the simulation results.

PHASE 4: (objective 4)

This phase aims to show our proposed solution performance by adapting it to work in a practical application.

Task 1: Studying the practical application requirements.

Task 2: Practical application development and evaluation.

Approach: Adapting the modified-TinyDDS to fit the practical application requirements.

3.2 Data security using cryptographic algorithms

Recently investigators have examined the effect of security algorithms on the energy consumption for WSNs. A number of [58-61] studies have found that using asymmetric-key algorithms consume more energy than symmetric-key algorithms in WSNs. Despite using methods like elliptic curve cryptography or dedicated cryptography coprocessors, which are considered as energy efficient methods, but the energy consumption overhead cost is still high.

In this work, we are interested in studying and applying block cipher symmetric-key algorithms to the TinyDDS middleware and reporting the overall performance evaluation in the network. We are considering the recommendations from previous studies while selecting suitable cryptographic algorithms.

Jinwala et al. [62] argue that using the block cipher XXTEA may be regarded as the optimal choice for data confidentiality in WSNs alongside applying the offset codebook (OCB) mode for authentication functionality. They have studied three algorithms, namely, Skipjack, AES, and XXTEA. They compared their performance on the Mica2 wireless sensor node platform using Avrora simulator. Likewise, Law et al. [63], studied the effect of block cipher algorithms in WSNs. They reported the influence of Microcontroller Unit (MCU) cycles and memory occupancy, but they did not report the energy consumption. Their study included several blocker cipher algorithms, such as Skipjack, RC5, RC6, MISTY1, Rijndael, Twofish, KASUMI, and Camellia. They recommend using Rijndael for security priority and MISTY1 for memory storage, and both are energy efficient among

the others. However, they did not provide details of the platform used to conduct their experiments.

Lee et al. [64] provide details regarding performance evaluation of block cipher algorithms and operation modes. They have analyzed four algorithms, AES, RC5, Skipjack, and XXTEA. Then, they have reported the influence of the key size for these algorithms on the energy consumption. They point out that their results are more realistic due to the use of actual sensor platforms. That is, by using Tektronix MSO 4034 oscilloscope, they are able to measure the drop in voltage, and then, find the energy consumption of the algorithms while considering their execution time. Their findings have been observed from conducting their experiments on two widely used platforms, MicaZ and TelosB; the same wireless sensor motes we are considering. Their results show that Skipjack and XXTEA are better in performance as they require fewer memory and consume less energy than the other algorithms. On the other hand, RC5 and AES provide more security than the others. The key size of RC5 and AES has a direct impact on energy consumed, as more execution rounds are needed for longer key sizes. Similarly, Trad et al. [65] investigate AES, RC5, and RC6, but they use Mica2 platform. Their results show that RC5 is the best cipher algorithm with repect to excecution time and the energy consumed.

A performance evaluation comparison study for WSN [66] that includes AES, RC6 and Scalable Encryption Algorithm (SEA), shows that AES is the best choice regarding the execution time, SEA requires the least memory storage, and RC6 performs the best regarding the bandwidth usage. Another performance comparison study for block cipher algorithms is analyzed by Biswas et al. [67]. The study covers Skipjack, XXTEA, RC5, AES, and CGEA and implements them on two platforms, Mica2 and Arduino Pro motes.

The results show that RC5 requires the least amount of memory storage, whereas Skipjack requires the most. However, Skipjack consumes the least energy in comparison with the other algorithms. AES and XXTEA consume less energy than RC5. On the other hand, AES with size key 128 is more secure than XXTEA, RC5, and Skipjack.

More recent attention has focused on proposing and developing lightweight block cipher algorithms for WSNs. For example, HIGHT [68], Simple Lightweight Encryption Scheme [69] and Lightweight Security Protocol [70] provide a good level of security, and they are considered energy efficient algorithms when examined on Mica2 platform. However, experimental results indicate that many of the lightweight algorithms have poor performance compared to conventional block ciphers [71, 72].

Relying only on encryption for data confidentiality without ensuring data integrity and authentication has been proven to be insecure [73]. There are different ways to implement message authentication code (MAC) to support data integrity. It can be constructed directly from block cipher algorithms using different approaches resulting in MAC variations such as OMAC, PMAC, CBC-MAC, XMAC, and CMAC. CBC-MAC is found to be insecure for long messages. The insecurity problem is resolved by using XMAC which operates on variable-length messages. CMAC is recommended by the National Institute of Standards and Technology (NIST), which also resolves the problems of CBC-MAC. Unlike XMAC, CMAC can operate with only one key [64]. Alternatively, HMAC is a technique using to construct a MAC without the need for cipher algorithms. The implementation of HMAC is based on cryptographic hash functions such as MD5 and SHA1 [51]. NIST considers HMAC-SHA1 [74] as a standard for generating HMAC. HMAC can be used to ensure data integrity and authentication at the same time. There are other implementations of MAC

algorithms based on universal hash functions which are considered very fast such as, UMAC [75] and VMAC [76].

Our implementation to support integrity and authentication considers the use SHA1 to construct HMAC. Alternatively, we used MMH [77] to construct the MAC based on the idea of universal hash functions. The reason for using MMH is that the MMH memory requirement is small as compared to SHA1. Moreover, the MAC size obtained from HMAC-SHA1 is 20 bytes, whereas, using MMH, the size of the MAC is only 4 bytes. A third approach that is considered to support integrity and authentication is based on MAC truncation. The MAC truncation method is used in WSNs to improve performance without the risk of losing security due to collisions [73]. Thus, it can be applied to produce 4 bytes MAC from 20 bytes HMAC-SHA1. In comparison, the size of the input block for SHA1 is 64 bytes, whereas MMH operates with variable block size. Hence, we consider implementing both options for HMAC; 20 bytes and truncated 4 bytes.

3.3 Performance Metrics

In this section, we present the performance parameters used to investigate the effect of adding security components for the middleware. Different scenarios for the network traffic will be constructed to evaluate the system behavior with respect to the memory footprint, energy consumption, and communication overhead performance metrics.

3.3.1 Memory Occupancy

This metric is evaluated by compiling the code of the middleware and the application on top of the operating system TinyOS. The memory occupancy shows how much is required before installing the code into the wireless sensor mote’s memory. The memory details of

ROM and RAM will be provided after the compilation of the code is successfully done. In our test, we will find the memory occupancy for four motes, MicaZ, Mica2 TelosB, and IRIS. Table 3 provides the on-board memory distribution of different motes.

Table 3: Memory size detail for different motes

Mote	RAM (kB)	ROM (kB)	EEPROM (kB)
MicaZ	4	128	512
Mica2	4	128	512
TelosB	10	48	1024 (external)
IRIS	8	128	4

3.3.2 Energy Consumption

Sensor nodes usually work on batteries which are considered a limited source of power. Hence, energy consumption metric is critical in WSNs. Therefore, it is necessary to maximize the lifetime of the sensors by trying to keep the communication and computation activities working with minimum power requirement. Online Energy Model (OEM) [78] is a tool based on POWERTOSSIMZ [79] extension, which is integrated into TOSSIM [57] simulator to support energy measurements for TinyOS applications. The energy consumption metric is calculated by using OEM for computing the total energy consumption for the entire network nodes. There are two main components contribute to the energy consumption, Radio and Microcontroller (MCU). Each component's value is calculated separately by taking the summation of energy consumption of all nodes in the network in the milli-Joule unit.

3.3.3 End-to-End Delay (EDD)

Latency or End-to-End Delay is evaluated by taking the average delay of the successfully received packets in the network. Although it depends on the underlying protocols in the middleware, in our comparison these protocols are fixed for all versions of the middleware

modifications. TOSSIM [57] simulator is used to track the packets from the moment they are sent from the Publisher's node side until they are successfully received at the Subscriber's node side. Each packet delay includes the packet processing, buffering and communication time.

3.3.4 Packet Delivery Ratio (PDR)

Packet Delivery Ratio is the percentage obtained by dividing the total number of successfully received packets at the Subscriber's node side by the total number of sent packets at the Publisher's node side. This metric gives an indication of the network behavior regarding the packet loss. Similar to EED metric, the PDR metric depends on the underlying protocols. The reliability QoS of the middleware is set to *best effort* option for all versions of the middleware modifications so as to compare between all these middleware versions. Note that if the QoS is set to be *reliable* in the middleware, the value of PDR is always one.

3.4 Simulation Tool

The main simulation tool used in our performance evaluation is TOSSIM [67], an event-driven simulator. According to [67] it is the most accurate simulator to simulate the behavior of WSN implemented on top of the TinyOS. It is written using network embedded systems C (nesC) [37] language, a dialect of C language. The energy performance metric is estimated using a plugin tool OEM [78] which is based on POWERTOSSIMZ [79] extension, integrated into TOSSIM. Currently, the current implementation of TOSSIM simulator only support simulating the behavior of only the MicaZ sensor nodes. Therefore,

our results for most of the performance metrics are available for only the Micaz sensor node. However, The OEM tool allows estimation of the energy for other sensor platforms, such as TelosB. Tables 4 and 5 provide the readings of the current consumption of Radio and Microcontroller (MCU) for the main components modes responsible for the energy consumption from the datasheet of both the MicaZ and TelosB sensor platforms. The OEM tool, estimate the energy consumption by substituting the value of the current for the corresponding mode. There are four modes or states for the Radio, Receive, Transmit, Idle, and Sleep. For the MCU, there are three modes, Active, Idle, and Sleep. The mode changes during the simulation. Accordingly, the value of the current changes, which impact on the total estimated energy. The energy consumption value depends on the value of the voltage value, the corresponding mode's current value, and the duration of time the sensor mode holds.

Table 4: Radio current consumption of MicaZ and TelosB

MicaZ		**TelosB**	
Mode	**Current**	**Mode**	**Current**
Receive	19.7 mA	Receive	23 mA
Transmit	17.4 mA	Transmit	17.4 mA
Idle	20 uA	Idle	21 uA
Sleep	1 uA	Sleep	1 uA

Table 5: MCU current consumption of MicaZ and TelosB

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

The default current readings in the OEM tools are for MicaZ platform. The current readings values of TelosB are applied in the OEM, replacing the values of MicaZ. Accordingly, we

can retrieve the estimated values of the energy consumption for Radio and MCU for TelosB sensor platform.

The OEM tool consists of several components. The values in the tables are added in the simulator files with the associated sensor mode. The reposnsible components for estimating the energy consumption readings for Radio and MCU in the simulator are *TossimPacktModelM* and *SimSchedularBasic,* repectively.

CHAPTER 4

IMPLEMENTATION OF SECURE TINYDDS

In this chapter, the implementation of adding security measures for middleware TinyDDS is explained. Several common security algorithms used in information security are considered for the implementation of the modified TinyDDS version. The new structure of the middleware components for the modified TinyDDS will be demonstrated to show how the security components are integrated into the basic implementation of TinyDDS.

4.1 Security Components

A considerable amount of literature has been published on evaluating the performance of algorithms to secure data in general. Our interest is for those algorithms that are considered lightweight and suitable for WSNs as aforementioned in the previous chapter. Some of those algorithms are implemented in our work as a Security Service component, and are expected to add overhead to the basic TinyDDS middleware. In the implementation of the Security Service component, we aim to provide security transparency concerning the application development. Hence, the security service is included as an integrated part of the middleware. Therefore, there is no need for the WSN applications developers to redesign or recode their applications. This allows the normal users with minimum knowledge of security functionality to secure the data indirectly through the use of the middleware. Accordingly, we target the support of data confidentiality and data integrity

through both lightweight and conventional cryptographic algorithms. Figure 5 depicts the extended architecture of TinyDDS after supporting the security services.

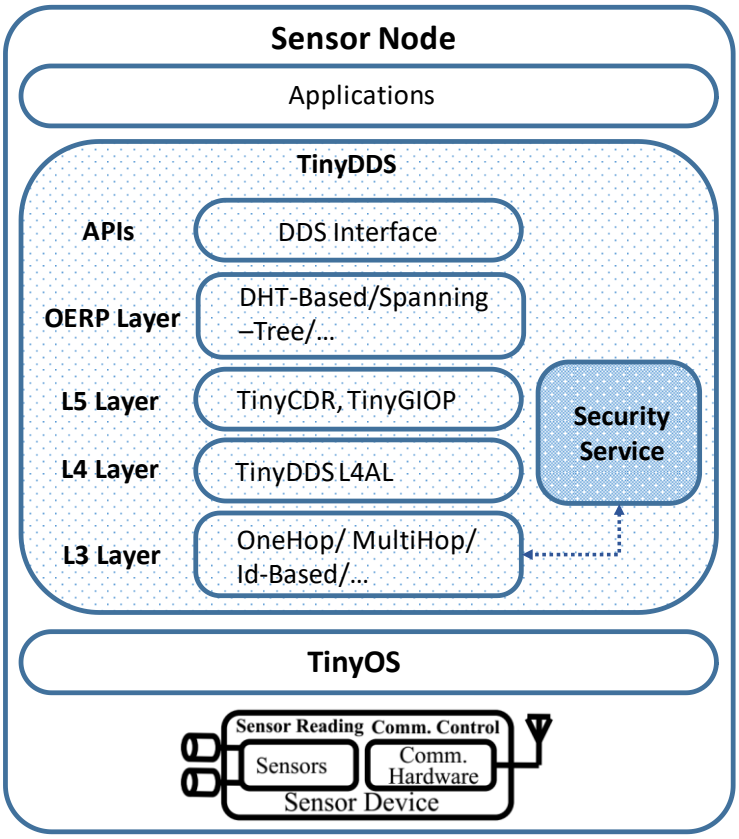


Figure 5: Extended TinyDDS architecture

The extended architecture for TinyDDS accounts for security through the introduction of the Security Service component. In the Security Service component, we have two main interfaces, *Secure_Info* and *UnSecure_Info*. The *Secure_Info* interface is invoked at the sender node to secure the data received from the L3 Layer of the TinyDDS by applying a selected cryptographic algorithm that is implemented in the modified middleware. The first task in the *Secure_Info* interface is applying the authentication process that includes exchanging the keys between…

and verifying their validity through checking the keys expiration time. The authentication process is explained in detail in subsection 4.4. The next task involves securing the packet through one of the confidentiality algorithms supported in our middleware. These algorithms have different characteristics that can change the way how the data is encrypted, especially the block size. The last step is to prepare the integrity requirements for the encrypted data by applying one of the selected integrity algorithms discussed in subsection 4.3. The necessary parameters are prepared for calculating the MAC to be compared later by the intended receiver node of the communication. At the receiver node, the *Unsecure_Info* interface is invoked. First, it will decrypt the received message, and then calculate the MAC from the decrypted data and compare it with the received MAC. It will accept the data and pass it to the L3 Layer of the TinyDDS if there is a match. Otherwise it will discard it.

4.2 Data Confidentiality requirement

In data confidentiality, we need to ensure that only authorized parties may have access to the exchanged data. The tool that is commonly used to achieve confidentiality is encryption. In our implementation, we attempt to construct a secure TinyDDS while considering the WSNs regarding source constraints of memory and energy.

We have considered a number of the block cipher algorithms for inclusion in the middleware according to three factors: memory, energy consumption, and security. The energy consumption is mainly caused by the computational complexity of the algorithm.

Table 6 provides the main parameters for some of suitable candidates for block ciphers algorithms discussed in the previous chapter.

Table 6: General parameters of block cipher algorithms

Algorithm	Key size (bits)	Block size (bits)	Rounds
AES [52]	128/192/256	128	10/12/14
Skipjack [50]	80	64	32
XXTEA [80]	128	64	14
RC5 [47]	128	64	14
PRESENT [81]	80/128	64	31
PRINCE [82]	128	64	12
SPECK [83]	64/72/ 96/ 128/144/192/256	32/48/ 64/ 96/128	26/42

Table 7 shows, in general, the main factors to compare between the block cipher algorithms. According to Table 7, to achieve the highest security in data confidentiality in a middleware AES should be selected. Whereas, if the main concern is saving memory space, the SPECK should be selected. Finally, SkipJack is the best choice if the energy consumption is concerned. The main implementation of our modified TinyDDS is constructed using AES to achieve the maximum security. Intensive simulations and scenarios are discussed in detail in Chapter 5.

Table 7: General comparison of block cipher algorithms

Algorithm	Memory	Energy consumption
AES	High	Medium
Skipjack	High	lowest
XXTEA	Low	Medium
RC5	Medium	High
PRESENT	Medium	Medium
PRINCE	High	Medium
SPECK	Lowest	Medium

4.3 Data Integrity requirement

Data integrity, we need to ensure the detection of unauthorized modification of the exchanged data. In many cases it is considered to be more critical than confidentiality. In our implementation to support integrity, we select two algorithms, SHA1 which is the standard algorithm to construct HMAC, and MMH for its relatively small memory occupancy.

The construction of the HMAC follows the standard HMAC-SHA1 by NIST [74], which requires to pass the SHA1 hash algorithm twice. There is secret key, to support the authentication of the message as well. The input block size of SHA1 is 64 bytes and the output is 20 bytes. The construction of the MAC from HHM algorithm requires a secret key for authentication, both the secret key and message have the same length. The size of the input block for HHM algorithm is variable, and the size of the output is 4 bytes.

4.4 Authentication and key exchange requirements

The modified middleware depends on the use of symmetric cryptographic algorithms to achieve confidentiality. Furthermore, the message authentication is applied through the use of HMAC integrity which depends on the presence of a shared secret key. Accordingly, we need a mechanism to exchange the shared secret keys with each session to prevent the reuse of the same key. To facilitate the sharing of a secret key, we implemented a key exchange protocol inspired by the Kerberos protocol [84]. The advantage of using the devised key exchange protocol is that the total number of required keys to securely exchange data

between any pair of nodes in the network is equal to the number of network. By comparison, the total number of keys needed when using a typical symmetric key cryptographic algorithm is $O(N^2)$, where N is the number of the network nodes. However, the devised protocol depends on the existence of a trusted-third-party (TTP) that manages the exchange of authentication messages between the nodes in the network.

The base-station node in WSNs is assumed to be the TTP. The publishers' nodes contact the base-station to establish mutual authentication and to obtain shared symmetric session key. Every node in the network is assumed to have its private key, which is only shared with the base- station. Moreover, the TTP has a private key that is not shared with any other node. The nodes' private keys are only used to achieve mutual authentication with the base-station node, and establish shared session keys. The session keys have an expiration time to force the need for establishing new session keys. The session keys are used to apply the confidentiality and integrity when exchanging the data messages between the nodes.

Figure 6 summarizes the devised protocol. Specifically, the first message in Figure 6 (a) shows a publisher (node 1) requesting the issuance of an identification ticket (TKT-I) from TTP (node 0). The identification ticket (TKT-I) is used later on to authenticate the publisher (node 1) to the TTP (node 0) when the publisher (node 1) wishes to contact either the TTP (node 0) as shown in Figure 6 (b), or a subscriber (node x) as shown in Figure 6 (c). When requesting the issuance of an identification ticket (TKT-I), the publisher (node1) sends its ID, '1', and a timestamp, t_1. The ID '1' is used by the TTP (node 0) to identify the private key, K_1, for the publisher (node 1). On the other hand, the timestamp t_1 is used

to prevent replay attacks [85]. Subsequently, the second message in Figure 6 (a) is sent from the TTP (node 0) to the publisher (node 1) which carries the issued TKT-I along with the encryption of the TTP ID, '0', the received timestamp, t_1, and the established session key, S_1. The encryption uses AES and the publisher's (node 1) private key, K_1, as identified by the TTP (node 0). The issued TKT-I is constructed by encrypting the publisher ID, '1', the established session key, S_1, and the expiration time for the established session key, t_{s_1}, using the TTP private key, K_{TTP}. Note that TKT-I can only be decrypted by the TTP (node 0). Note also that TKT-I contains all necessary information to identify to whom the ticket was issued and what was the established session key, S_1, between the TTP (node 0) and the publisher (node 1). Hence, once TKT-I is issued and sent to the publisher (node 1), the TTP (node 0) can discard the issued TKT-I. Accordingly, the TTP conserve its memory by discarding the issued TKT-I once delivered to the intended publisher.

Once the publisher (node 1) receives the second message in Figure 6 (a), it can then use the received TKT-I and $S_1$to contact either the TTP (node 0) or a subscriber (node x) as shown in Figure 6 (b) and Figure 6 (c), respectively. Figure 6 (b) shows how mutual authentication is achieved between the TTP (node 0) and the publisher (node 1). Specifically, the first message is sent by the publisher (node 1) that carries TKT-I and the encryption of a newly generated timestamp, $t_{1'}$, using the established session key, S_1. Once received, the TTP (node 0) decrypts TKT-I to extract the established session key, S_1, and uses it to decrypt and extract the publisher's (node 1) timestamp, $t_{1'}$. Subsequently, the TTP (node 0) encrypts its ID, '0', a newly selected timestamp, $t_{0'}$, and decrypted and extracted the publisher's (node 1) timestamp, $t_{1'}$, using the established session key, S_1. Once received, the publisher (node 1) authenticates that the sending side is the TTP (node

0) as the publisher's (node 1) own timestamp, $t_{1'}$, is included in the received message. Accordingly, the publisher (node 1) sends the third message to TTP (node 0). The third message is constructed by encrypting the publisher's ID, '1', and the TTP timestamp, $t_{0'}$, using the established session key, S_1. Once received, the TTP (node 0) authenticates the sending side is the publisher (node 1) as the TTP's (node 0) timestamp, $t_{0'}$, is included in the received message.

Similarly, Figure 6 (c) shows how mutual authentication is achieved between the publisher (node 1) and the subscriber (node x) with the help of TTP (node 0). Precisely, the publisher (node 1) send the first message to the TTP (node 0) to request the issuance of a granting ticket (TKT-II). The first message carries the ID of the publisher, '1', the ID of the subscriber, 'x', TKT-I, and the encryption of a newly generated timestamp, t_1^* using the established session key, S_1. Once received, the TTP decrypts TKT-I to extract the established session key, S_1, and uses it to decrypt and extract the publisher's (node 1) timestamp, t_1^*. Accordingly, the TTP (node 0) issues TKT-II. Moreover, it encrypts the TTP ID, '0', a newly established session key between the publisher (node 1) and the subscriber (node x), K_{1x}, the publisher's timestamp, t_1^*, and the expiration time for the established session key between the publisher and the subscriber, $t_{s_{1x}}$, using the extracted session key, S_1. Both TKT-II and the result of the encryption are sent by the TTP (node 0) to the publisher (node 1) in the second message. TKT-II is constructed by the TTP (node 0) by encrypting the publisher ID, '1', the newly established session key, K_{1x},and the expiration time for the established session key, $t_{s_{1x}}$, using the subscriber private key, K_x. Note that aside from the TTP only the subscriber (node x) can decrypt TKT-II. Once received, the publisher (node 1) authenticates the TTP (node 0) as the publisher's

timestamp, t_1^*, is included in the encrypted part of the message. Accordingly, the publisher extracts the established session key, K_{1x} . At this point, the publisher (node 1) can use both TKT-II and the established session key, K_{1x} , to contact the subscriber (node x). Thus, the third, fourth, fifth messages presented in Figure 6 (c) serve the same purpose as those presented in Figure 6 (b).

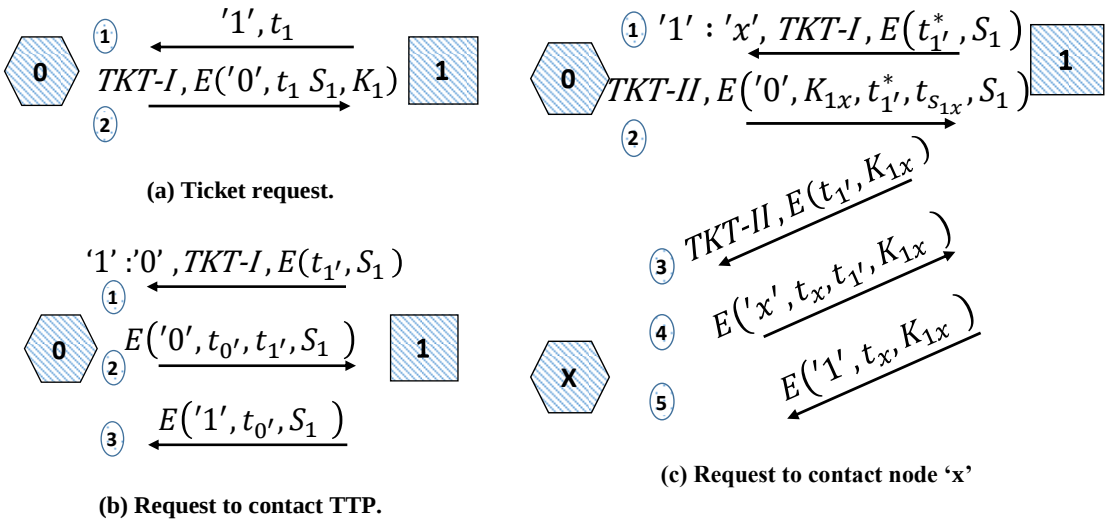


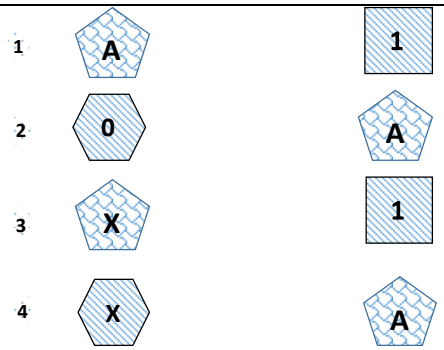
Figure 6: Messages flow of the protocol for authentication and session key exchange

Figure 7 shows two common types of attacks in WSNs. Figure 7 (a) presents the Man-in-the-middle attack, and Figure 7 (b) presents the Impersonate attack. Figure 7 (c) shows the symbols represented in the network. Our devised protocol for mutual authentication and key exchange is immune to these types of attacks. We will refer to Figure 6 and which message number the attack fails for all possible scenarios in the two attacks explained next.

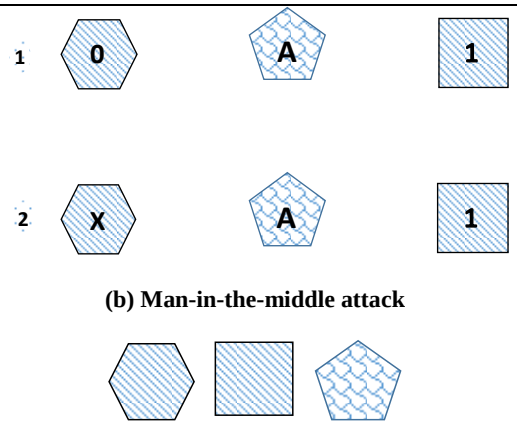
Regarding the Impersonate attack, a node in the network pretends to be another node in the network. There are four possible scenarios for the attacker node. Figure 7 (a) shows the four cases associated with a label number. The first case, the attacker node A, pretends to

be the TTP node. The attacker fails at message number 2 in Figure 6 (a), by failing to send back the encrypted timestamp, t_1, since only node 1 and node TTP share the secret key, K_1, as shown in Figure 7 (d). In the second case, the attacker node A, pretends to be a Publisher node and tries to be authenticated as node 1 for instance, as shown in Figure7 (a.2). The attacker fails at message number 1 in Figure 6 (b), by failing to send the encrypted newly generated timestamp $t_{1'}$ with the common session key S_1, which is part of TKT-I, as shown in Figure 7 (e). The third case, the attacker pretends to be a Subscriber node, as for example node x. The attacker fails at message number 4 in Figure 6 (c), by failing to send back the encrypted $t_{1'}$ with the common session key K_{1x}, that only node 1 and node x share, as shown in Figure 7 (e). The last case, the attacker node pretends to be a Publisher node, as for example node 1. The attacker fails at message number 3 in Figure 6 (c), by failing to send the encrypted $t_{1'}$ with the common session key K_{1x} , which is part of TKT-II that only node 1 and node x share, as shown in Figure 7 (f).

Regarding the man-in-the-middle attack, there are two possible cases for the attacker node. Figure 7 (b) shows the two cases associated with a label number. The first case, the attacker node is between the Publisher node and the TTP node. The attacker fails at message number 2 in Figure 6 (a), by failing to send back the encrypted t_1 with the key K_1, that only node 1 and node x share, as shown in Figure 7 (g). The second case, the attacker node is between the Publisher node and a Subscriber node, as for example node x. The attacker fails at message number 4 in Figure 6 (c), by failing to send back the encrypted t'_1 with the session key K_{1x}, that is included in TKT-II, and only node 1 and node x share, as shown in Figure 7 (h).

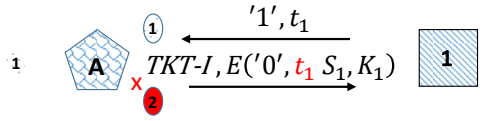


(a) Impersonate attack

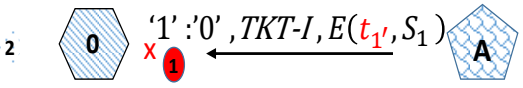


(b) Man-in-the-middle attack

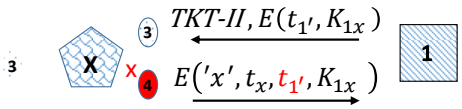
(c) Symbols from left to right: Subscriber, Publisher, Attacker node



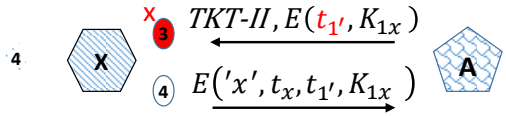
(d) Impersonate attack case 1



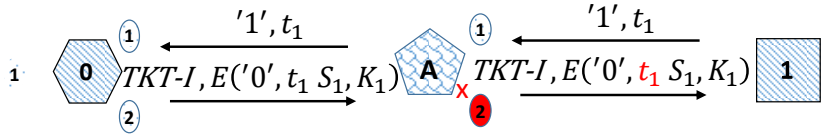
(e) Impersonate attack case 2



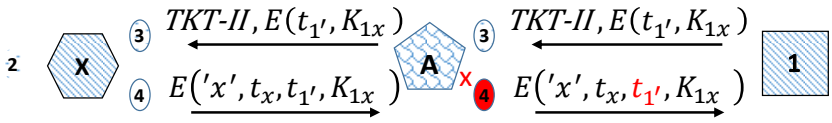
(e) Impersonate attack case 3



(f) Impersonate attack case 4



(g) Man-in-the-middle attack case 1



(h) Man-in-the-middle attack case 2

Figure 7 : Attack scenarios.

CHAPTER 5

SIMULATION RESULTS AND ANALYSIS

In this chapter, the simulation setup and parameters are provided. The proposed implementation of the modified TinyDDS with security integrated will be evaluated and compared with the basic TinyDDS results. Several parameter factors are considered in our simulations: The memory space occupation for different mote platforms, namely, Mica2, MicaZ, Iris, and TelosB. The energy consumption of the whole network is calculated by taking the summation of all nodes consumption. Moreover, the end-to-end delay and the packet delivery ratio are evaluated for packet transmission. The default QoS parameters are considered for the basic and the modified TinyDDS. Furthermore, the reliability QoS is set to *BEST_EFFORT*. Accordingly, reliable data transfer is not guaranteed.

5.1 Simulation setup and parameters

Several experiments are conducted to evaluate the performance of the middleware. The modified middleware is compared against the basic TinyDDS. Throughout this chapter, the term TDDS will be used to refer to the basic implementation of the TinyDDS. Table 8 illustrates some of the key parameters used in of the simulation. The setup of the simulation parameters is inspired from the work in [78]. The experiments are tested using TOSSIM [67] simulator over a MicaZ mote platform, the radio model of the MicaZ mote is based on Chipcon CC420 model.

Table 8: Simulation parameters.

Parameter	Value
Topology	Square grid
Area	100x100 Meter^2
Number of Nodes	25
Simulation time	500 seconds
Mote platform	MicaZ
Radio model	Chipcon CC420
Data rates	0.5, 1, 2, 4, 8 packet/second
Number of publishers	1, 2, 4, 8, 16, 24
Packet payload size	16, 32, 48 bytes
Data point reading	10 times

The application to be used in our experiments is simply collecting readings from nodes called Publishers. The flowchart of the application is depicted in Figure 8. The readings are aggregated locally at the publishing nodes, then the average of these readings is sent to other nodes, called Subscribers. In our scenario, we have only one Subscriber which is also considered to be the Base-station, and a variable number of Publishers. We study the effect of increasing the number of Publishers from 1, 2, 4, 8, 16 to 24 to evaluate the performance of the network and monitor the scalability. The network traffic load varies by changing the data rate. The data rate varies in our tests from 0.5 to 8 packets per second. When the data rate is 8 packets/second the traffic is expected to be very high, and the packet loss increases, Subsequently, the throughput becomes very low.

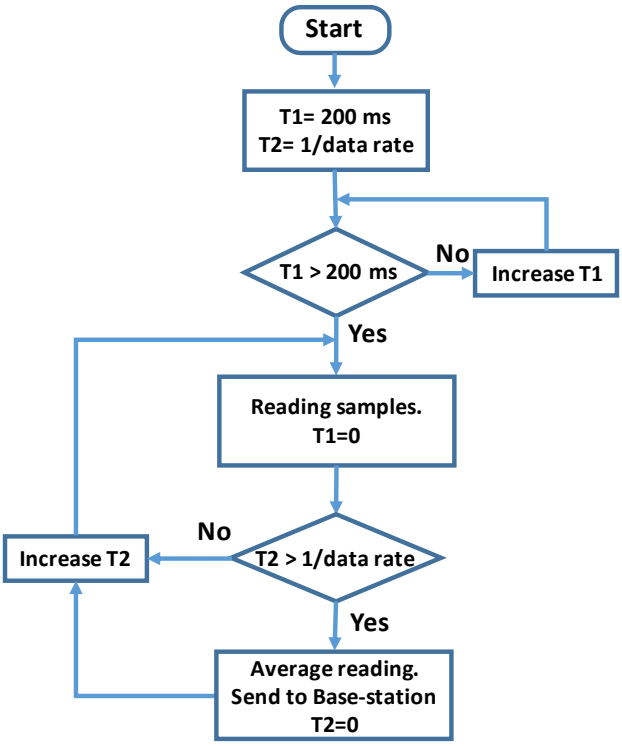


Figure 8: Application function in the middleware

The topology of the network considered in our experiments is a square grid topology to reduce the simulation time required for our test scenarios. The network topology is illustrated in Figure 9, where we have a square grid network of 25 nodes that are uniformly distributed in a 100x100 square meter area. The node at the lower corner with ID 0 is the only Subscriber and is considered to be the base-station, and the square-shaped nodes are the Publishers. The remaining nodes are relay nodes that are used for packet forwarding purpose according to the routing protocol. Different scenarios are considered for the number of Publishers in our experiments. The nodes are distributed in a way that the path from the Publisher node is as far or symmetric-shaped as possible to the Subscriber node. The distribution of the nodes is not critical as we are investigating the effect on performance for the secure and unsecure middleware cases using the same network configuration. The last network topology considered is the full duty network where one is

considered to be a Subscriber and the remaining nodes act as Publishers. The TinyDDS middleware follows DDS standards. Therefore, for simplicity, we have one topic value which the Subscriber is interested in, according to the pub/sub functionality. The topic represents the data that the sensor nodes are collecting, for example, temperature, pressure, humidity or GPS location. In our experiments, our topic value is temperature, but the middleware is reading it as a random value.

There will be multiple secured versions for the modified-TinyDDS middleware. We investigate a number of block ciphers (AES, SPECK and PRINCE) and hash algorithms (MMH and SHA1) according to the recommendations provided in the previous chapter. We consider the memory constraints, to enable the middleware to work normally with limited memory motes, such as Arduino Pro and TelosB. For all versions of secured TinyDDS, performance evaluation and comparisons with the basic TinyDDS are conducted.

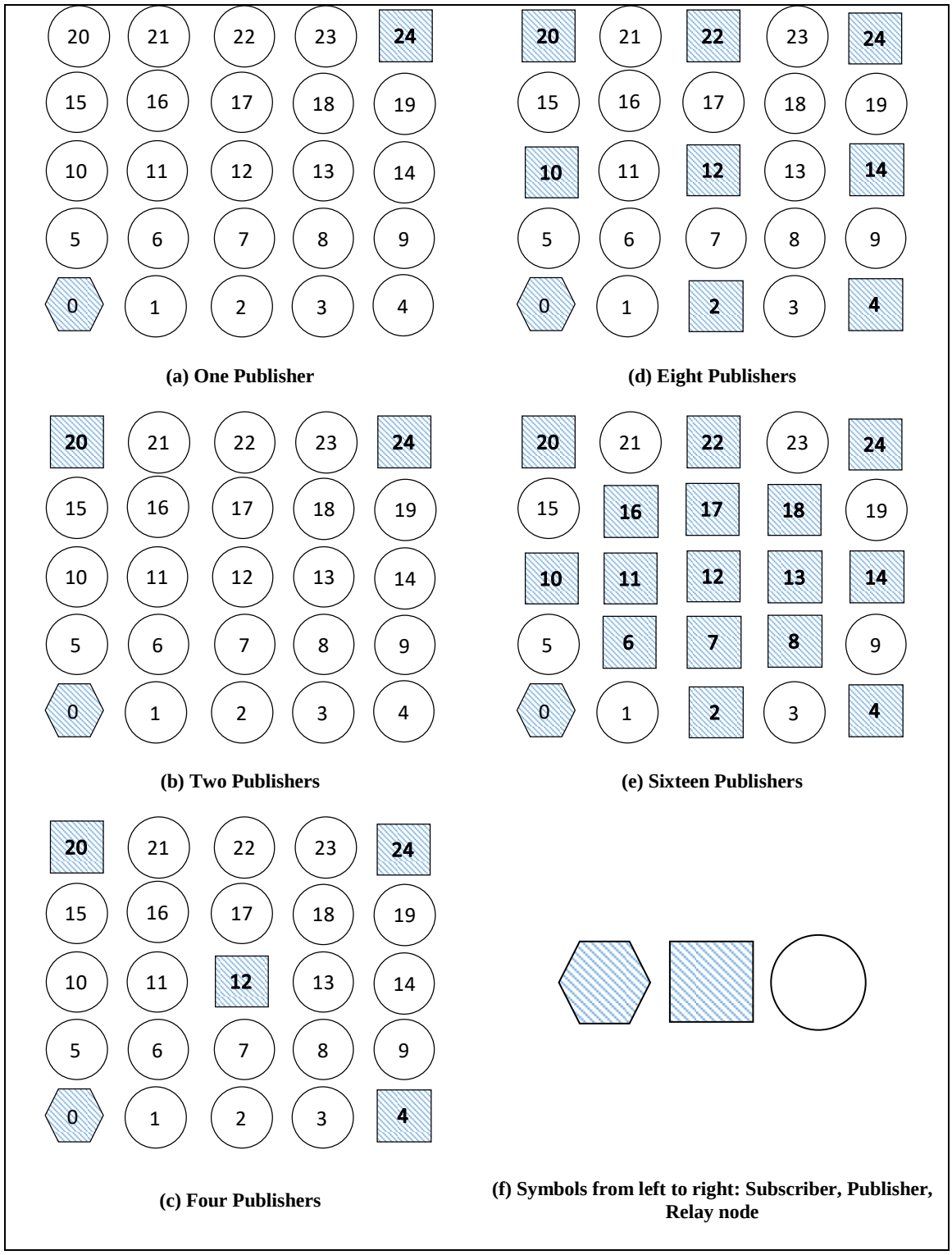


Figure 9: Different scenarios showing the network grid Topology with changing in number of publishers.

5.2 Simulation results and analysis

In our evaluation study to estimate overhead after adding security measurements to TinyDDS middleware, we perform several experiments by mainly using TOSSIM simulator and MicaZ mote platform. The main performance metrics considered are Memory occupancy of the middleware, Packet Delivery Ratio, Latency or End-to-end delay, and total energy consumption.

5.2.1 Memory Occupancy metric

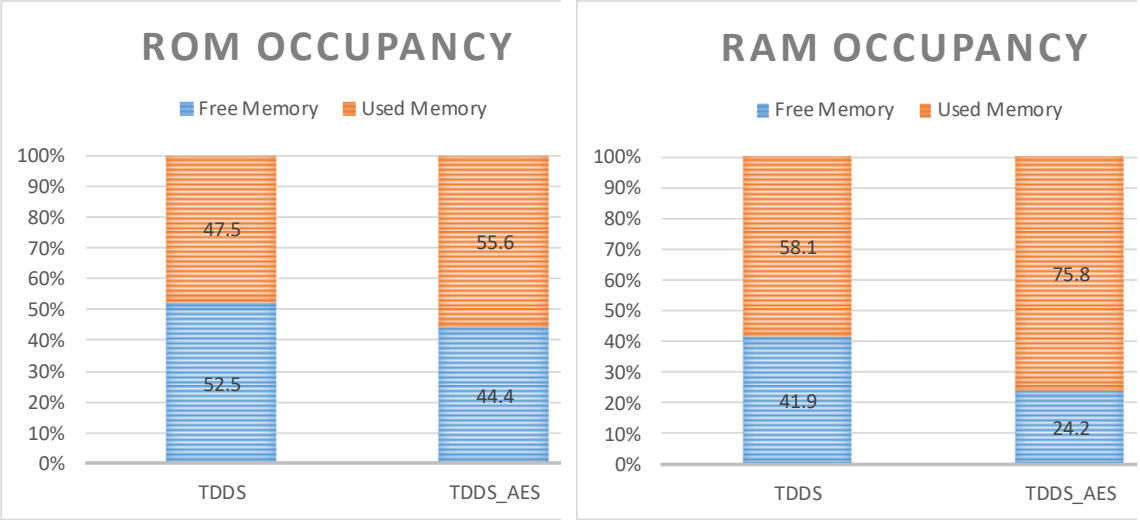
This metric is evaluated by compiling the code of the middleware, and the application tested on top of the TinyOS. The application is simply, reading a sensor value from a publisher node to be sent to the base station, i.e. the subscriber node. Two versions of the code for the middleware are prepared to study the effect of adding security measure. The first one is the basic TinyDDS, and the other one is the modified-TinyDDS with security measures. As stated earlier, there are multiple implementations of the modified-TinyDDS, to account for different security algorithms. The first security implementation adds the AES algorithm as component to ensure data confidentiality.

There are four mote platforms considered in this test. Namely, Mica2, MicaZ, IRIS, and TelosB. The memory amount requirement is different for each mote platform. The compilation process for preparing the executable image of the code, reveals the required amount of ROM, and RAM memory when installing the code to the mote.

The memory space for TelosB mote is 48KB and 10KB of ROM and RAM, respectively. Hence, Table 9 provides the full TelosB memory occupancy of the basic TinyDDS and the modified TinyDDS after adding the confidentiality using AES algorithm. It can be seen from the table that the amount of memory needed for the middleware and the testing application is 23324 bytes for the ROM and 5952 bytes for the RAM. Equivalently, they require a total requires 47.5% and 58.1% of the overall memory available in the mote for ROM and RAM, respectively. The percentage of the contribution after adding AES components is 8.1% of the ROM and 17.7% of the RAM. Hence, the percentage of the remaining memory is 44.4% of 48KB and 24.2% of 10KB for ROM and RAM, respectively. Furthermore, Figure 10 depicts the percentage difference of the memory occupancy between the basic TinyDDS and the modified-TinyDDS for TelosB mote.

Table 9: Memory occupancy for TelosB mote.

	Memory Occupancy			
	ROM (B)	**RAM (B)**	**ROM (%)**	**RAM (%)**
Application and MW	**23324**	**5952**	**47.5**	**58.1**
Adding AES	**4014**	**1810**	**8.1**	**17.7**
Remaining Memory	**21814**	**2478**	**44.4**	**24.2**
Total Memory	**49152**	**10240**	**100**	**100**



(a) ROM occupancy **(b) RAM occupancy**

Figure 10: Memory usage for Basic TinyDDS and Modified-TinyDDS based on TelosB mote with confidentiality.

The test code is also applied on three other motes. Table 10 is summarizing the memory results after applying the test. We notice that the MicaZ mote requires the most amount of ROM memory to install the test code, whereas TelosB requires the least amount of ROM memory space to install the code. However, TelosB requires the most amount of RAM memory to install the test code, whereas Mica2 requires the least amount of RAM memory space to install the code.

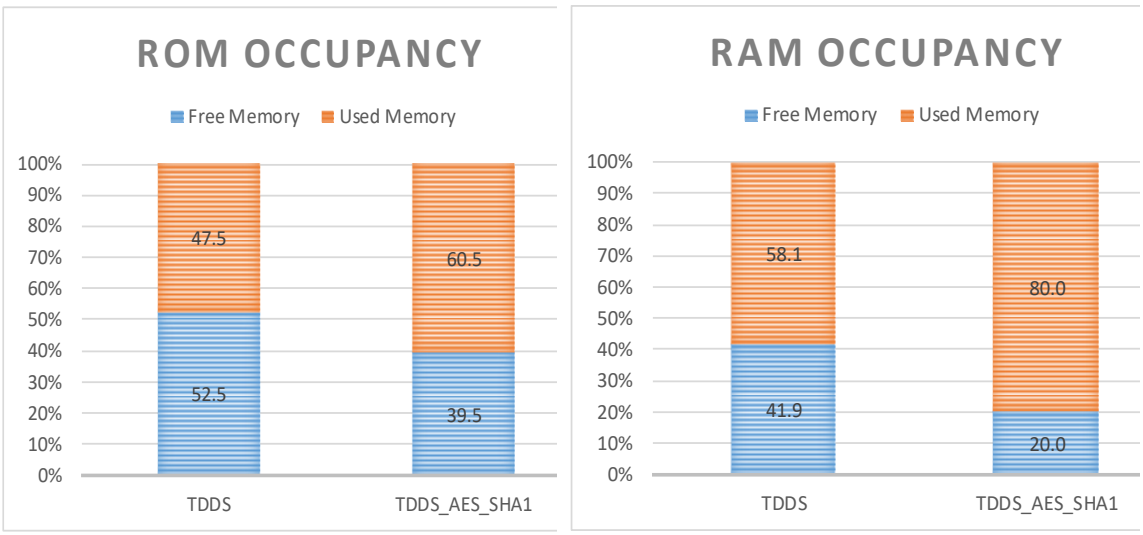
Table 10: Memory occupancy for different motes (size in bytes).

	Security Off (TDDS)		Security On (TDDS_AES)	
	ROM	RAM	ROM	RAM
MicaZ	30002	5806	35288	7616
Mica2	26734	5710	32350	7520
IRIS	28740	5897	33546	7709
TelosB	23324	5952	27338	7762

So far confidentiality has only been applied to the basic TinyDDS middleware. However, applying confidentiality without integrity has been proven insecure in previous studies. The first algorithm for integrity which has been investigated in our study is SHA1. Table 11 shows the detail distribution of memory occupancy after adding the SHA1 components to support the integrity measure. It is apparent from this table that the contribution to the ROM occupancy of adding integrity components with SHA1 algorithm is equal half of the contribution of adding confidentiality components with AES algorithm. The remaining memory is 39.5% and 20% for ROM and RAM, respectively. Furthermore, Figure 11 depicts the percentage difference of the memory occupancy between basic the TinyDDS and the modified-TinyDDS for TelosB mote after adding AES for confidentiality and SHA1 for integrity components.

Table 11: Memory occupancy for TelosB mote with confidentiality and integrity (SHA1).

	Memory Occupancy			
	ROM (B)	RAM (B)	ROM (%)	RAM (%)
App. and MW	23324	5952	47.5	58.1
Adding AES	4014	1810	8.2	17.7
Adding SHA	2376	434	4.8	4.2
Remain. Memory	19438	2044	39.5	20.0
Total Memory	49152	10240	100	100



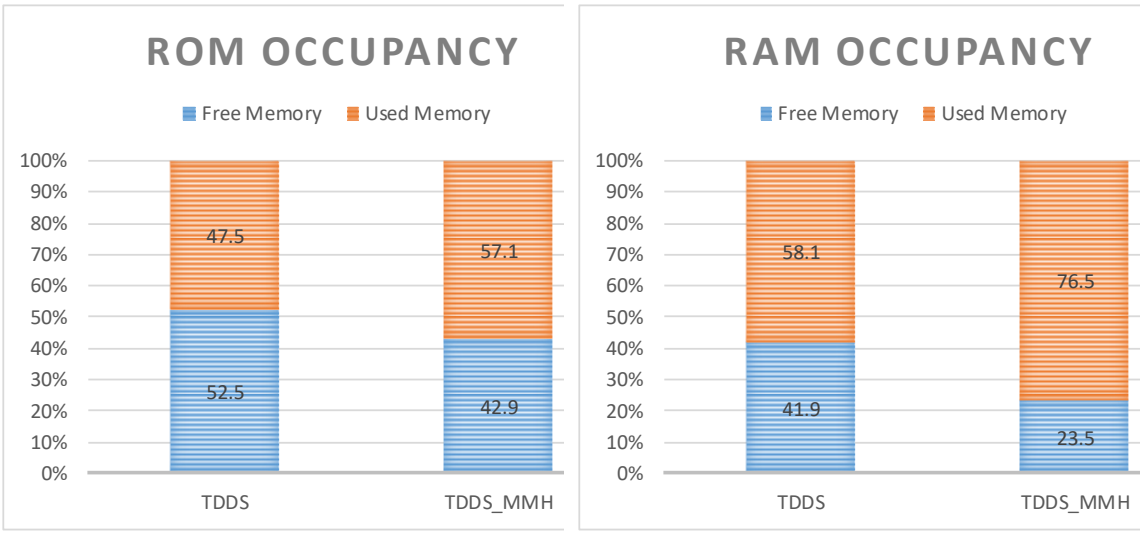
(a) ROM occupancy **(b) RAM occupancy**

Figure 11: Memory usage for Basic TinyDDS and Modified-TinyDDS based on TelosB mote with confidentiality and integrity (SHA1).

In similar steps, to make more memory space available, we have investigated the use of another algorithm to produce the MAC mechanism as aforementioned in the previous chapter. The integrity components are constructed by using the MMH algorithm. Table 12 provides an overview of the significant reduction in the memory occupation as a result of using the MMH algorithm. The contribution of using SHA1 is clearly larger than MMH. As can be seen from the data in the table that the percentage of memory required to add the MMH algorithm is 1.4% and 0.7% for ROM and RAM, respectively. On the other hand, the percentage of memory needed to add SHA1 is 4.8% and 4.2% for ROM and RAM, respectively. Figure 12 shows the percentage difference of the memory occupancy between the basic TinyDDS and the modified-TinyDDS for the TelosB mote after adding AES for confidentiality and MMH for integrity.

Table 12: Memory occupancy for TelosB mote with confidentiality and integrity (MMH).

	Memory Occupancy			
	ROM (B)	RAM (B)	ROM (%)	RAM (%)
App. and MW	23324	5952	47.5	58.1
Adding AES	4014	1810	8.2	17.7
Adding MMH	710	74	1.4	0.7
Remain. Memory	21104	2404	42.9	23.5
Total Memory	49152	10240	100	100



(a) ROM occupancy **(b) RAM occupancy**

Figure 12: Memory usage for Basic TinyDDS and Modified-TinyDDS based on TelosB mote with confidentiality and integrity (MMH).

5.2.2 Energy Consumption metric

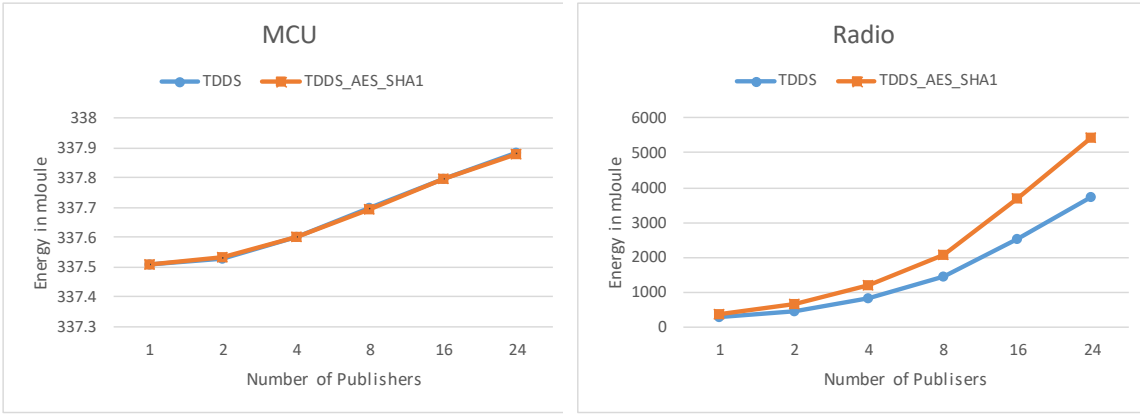
The energy consumption of the sensor nodes is one of the important metrics for WSNs performance evaluation. In our experiments, the energy consumption is calculated by taking the summation of the energy consumption of all nodes in the network in milli-Joule (mJ) unit, and it is obtained by using the OEM tool and TOSSIM simulator. The main

components that directly contribute to the energy consumption are the Radio and Microcontroller (MCU).

Following the same setup of the experiment's parameters as earlier, a comparison in performance behavior between the basic TinyDDS and the modified versions of TinyDDS that account for security measures are presented. Two versions of the modified TinyDDS are investigated. The main algorithm for our study is AES which achieves maximum security with respect to data confidentiality. Whereas the main algorithm to achieve maximum security with respect to data integrity is SHA1. Since SHA1 requires a considerable memory space as demonstrated in the previous section, MMH is investigated as well. We have studied the performance of the middleware in the simulation runs while changing the number of Publishers from 1 to 24 (full load). The Publisher node is responsible for causing data traffic.

Throughout this work, the term TDDS will refer to the basic TinyDDS, and the term TDDS_X_Y will refer to the modified TinyDDS with X as the confidentiality algorithm and Y as the integrity algorithm. Figure 13 provides a comparison of energy consumption between TDDS and TDDS_AES_SHA1, while changing the number of Publishers when the data rate is one packet/s. As shown in Figure 13 (a) for MCU, there was no significant difference between TDDS and TDDS_AES_SHA1. That is because the contribution of the security components for energy consumption in MCU is in the scale of micro-Joules (μJ). On the other hand, Figure 13 (b), shows a clear difference in the energy consumption in the Radio component as the number of the Publishers increases from 8 to 24, and a slight difference as the number of the Publishers increases from 1 to 4. The packet payload size for TDDS_AES_SHA1 48 bytes which is larger than the packet payload size for TDDS 16

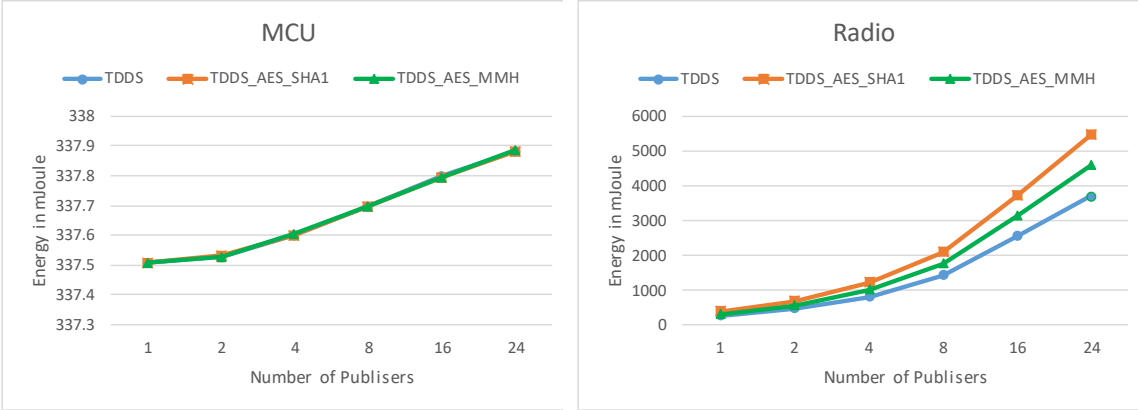
bytes. This is due to encryption and additional MAC bytes. As a result, the energy consumption for TDDS_AES_SHA1 is larger. Moreover, increasing the number of Publishers results in an increase in the traffic in the network.



(a) MCU **(b) Radio**

Figure 13: Energy consumption comparison between TDDS and TDDS_AES_SHA1 when the data rate is 1p/s.

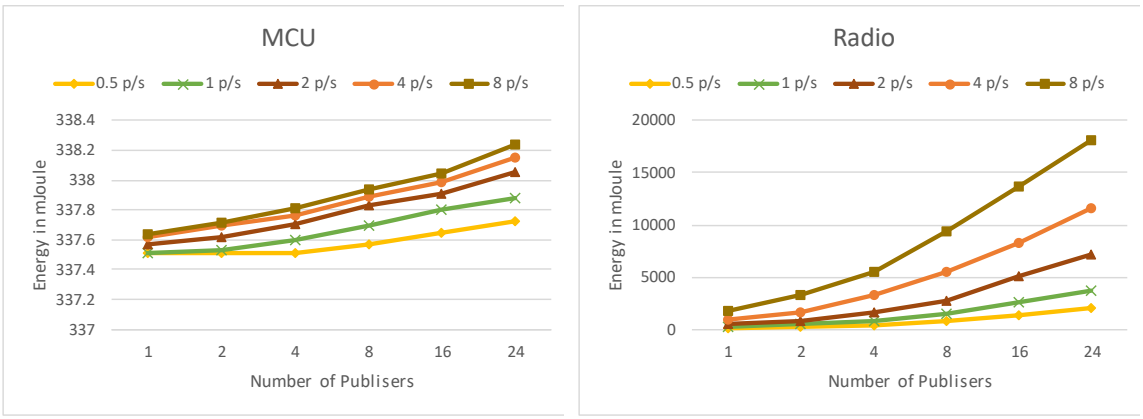
Similarly, Figure 14 provides a comparison of energy consumption when the data rate is 1p/s, but this time the TDDS_AES_MMH is added to the comparison. Figure 14 (a) has a similar MCU observation as the one obtained from Figure 14 (a). Also, from Figure 14 (b), we can see that using MMH as an integrity algorithm results in a reduction in the radio energy consumption as compared to using SHA1. The reason for that is the reduction in size of payload for the packet from 48 bytes when using SHA1 to 32 bytes when using MMH. Thus, using the MMH algorithm for integrity will save Radio energy.



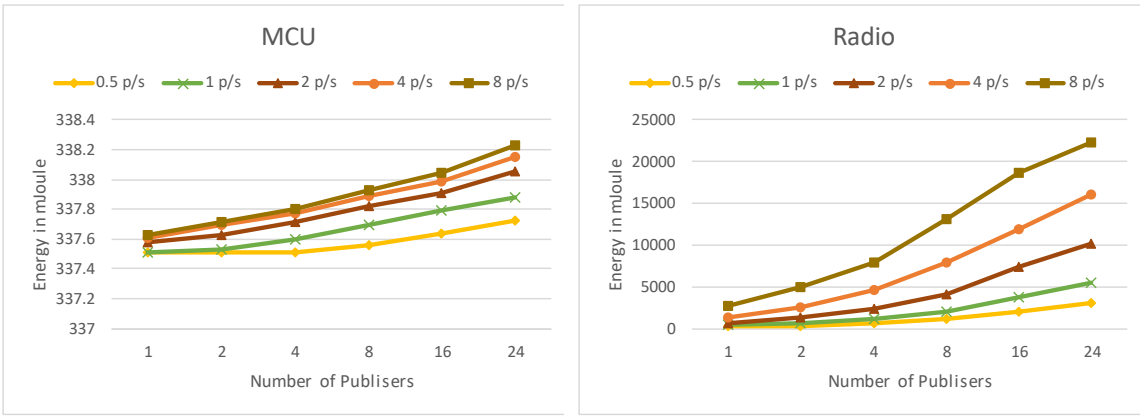
(a) MCU **(b) Radio**

Figure 14: Energy consumption comparison for all MW versions when the data rate is 1p/s.

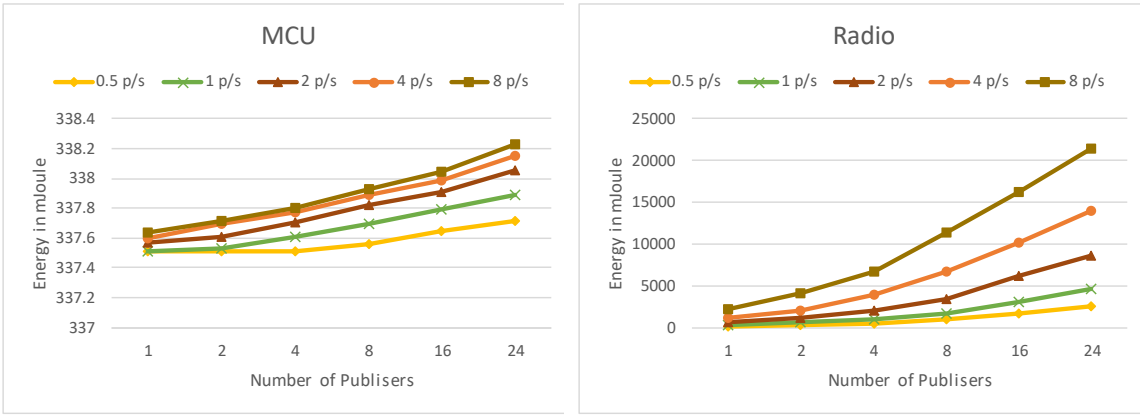
Furthermore, we conducted additional simulation runs where we change the data rate from 0.5 p/s to 8 p/s to study the effect of the traffic load in the network for all versions of the middleware. As shown in Figure 15, the MCU results are almost identical for all versions of the middleware as we observed earlier. Also, for the readings of each middleware version, there is a slight increase in the MCU energy consumption while the data rate changes from 0.5 p/s to 8 p/s. For the Radio energy consumption, the effect of applying different data rates is evident following the same trend for all middleware versions. In every data rate option, TDDS_AES_MMH consumes less Radio energy than TDDS_AES_SHA1.



(a) Results for TDDS: MCU and Radio



(b) Results for TDDS_AES_SHA1: MCU and Radio



(c) Results for TDDS_AES_MMH: MCU and Radio

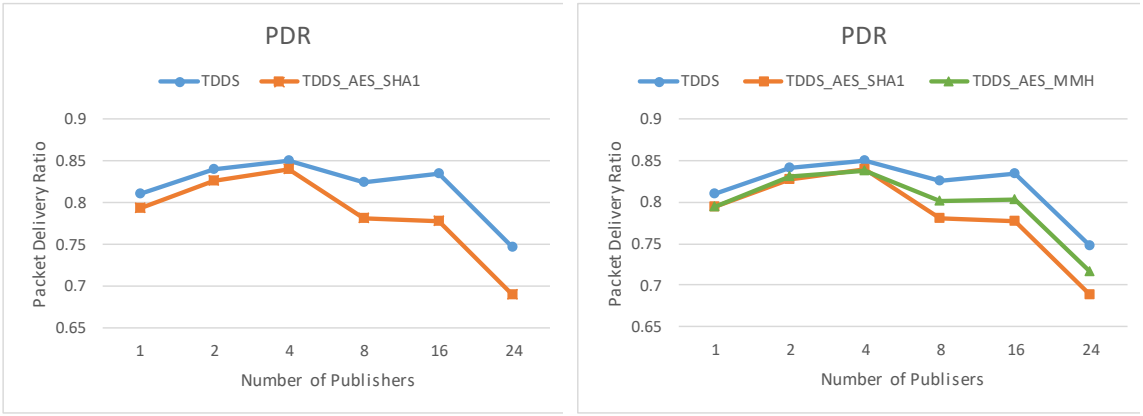
Figure 15: Energy consumption results for all MW versions with different data rates.

5.2.3 Packet Delivery Ratio (PDR) metric

Packet delivery ratio gives an indication of the system behavior towards stability. This metric is evaluated using TOSSIM simulator by dividing the total number of successfully received packets at the Subscriber node(s) by the total number of packets sent from the Publisher node(s). In all middleware versions, the QoS Reliability parameter is set to *BEST_EFFORT* in our simulation setup. Therefore, reliable data transfer of data is not guaranteed. Whereas, if the QoS Reliability parameter is set to *RELIABLE*, then the data will be reliably delivered to the Subscriber(s). As Such, the value of PDR will always be one, but this will result in extra overhead to the network traffic.

Figure 16 provides a comparison of the PDR value when the data rate is 1p/s. Figure 16 (a) shows that the PDR for TDDS is over 0.8 in most cases for all Publishers number except for when it is 24 (full load). Contrary to initial expectation, there is an increase in the PDR when increase from 1 to 2. This can be attributed to the location of the Publisher nodes for the two cases. In the case of 1 Publisher, the location is at the far end of the network, whereas the second case one of the two Publishers is closer to the Subscriber. This observation is also true for in the case of 4 Publishers, but with a very slight increase. Although there is a Publisher at the far end like case 1, the contribution of successfully received packets of the closer nodes to the Subscriber mitigates the remote nodes packet loss. The traffic congestion in the case of 8 Publishers and onward, starts to impact the PDR and the packet loss is at the highest point in the case of 24 Publishers. Similarly, TDDS_AES_SHA1 follows the same trend, but in every case the value of the PDR is lower than the TDDS PDR results. The size of the packet in TDDS_AES_SHA1 is longer than that of the TDDS, and that contributes to the packet loss. Figure 16 (b) shows the results

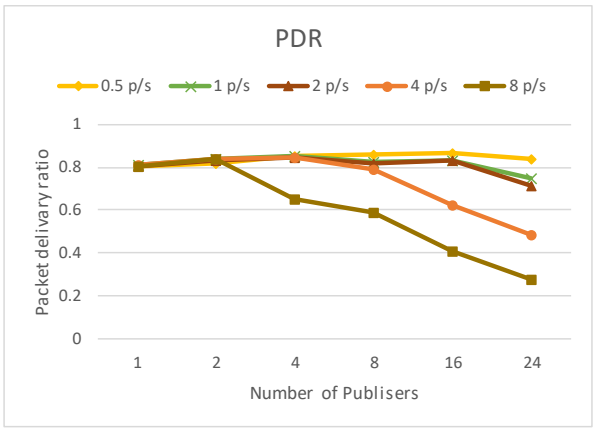
after adding TDDS_AES_MMH to the comparison. As expected, it follows the same trend, and performs better than TDDS_AES_SHA1 in most cases.



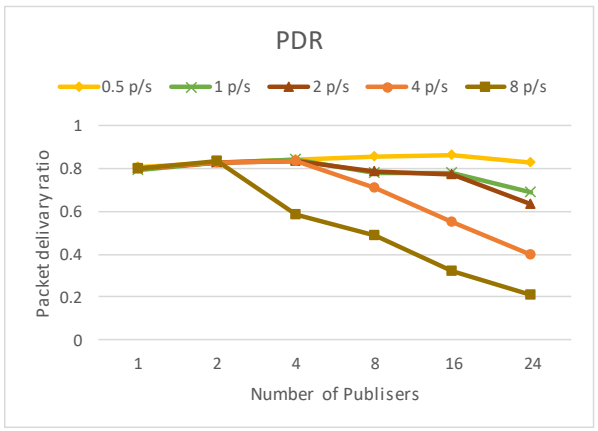
(a) TDDS vs TDDS_AES_SHA1 **(b) All three MW versions**

Figure 16: Packet delivery ratio comparison for all MW versions when the data rate is 1p/s.

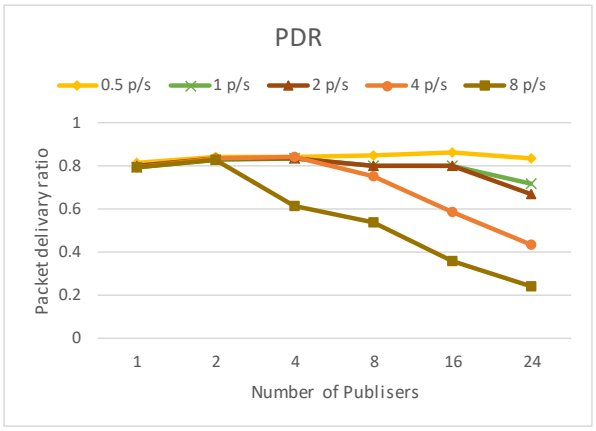
Figure 17 compares the results obtained from changing the data rate from 0.5 p/s to 8 p/s for all cases. It is clear from the figure that the change is almost constant when the data rate is 0.5 p/s for all cases. The PDR value decreases when the data rate increases and the packet loss is very high when the data rate reaches 8 p/s. In all Publishers cases, TDDS_AES_MMH performs better than TDDS_AES_SHA1, as expected.



(a) Results for TDDS



(b) Results for TDDS_AES_SHA1



(c) Results for TDDS_AES_MMH

Figure 17: Packet delivery ratio comparison results for all MW versions with different data rates.

5.2.4 End-to-End Delay metric

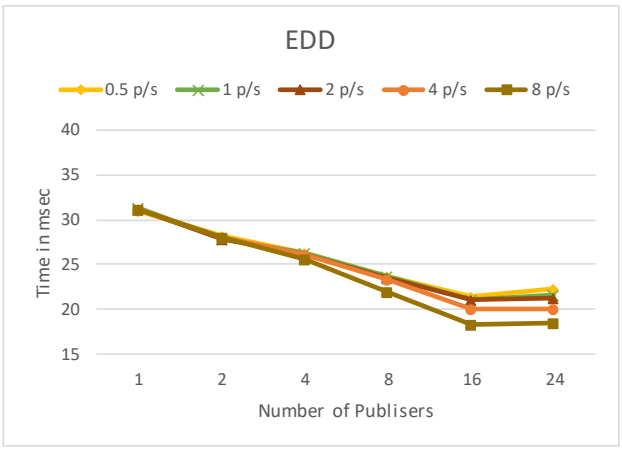
The End-to-End Delay (EED) metric is a good indicator for the impact of adding security measure on the WSN performance. The EED includes processing, transmission, and security computation delay of the packets. The EED is calculated by taking the average delay of all successfully received messages by the subscriber node.

Figure 18 provides a comparison of the EED value when the data rate is 1p/s. Figure 18 (a) shows the EED comparison between TDDS and TDDS_AES_SHA1. It is clear that the delay decreases as the number of Publishers increases from 1 to 16. The location of the Publisher impacts the EED delay. The farther the distance of the Publisher from the Subscriber the longer the time the packets need to arrive at the Subscriber. In the 1 Publisher case, the Publisher is at the far end in the network away to the Subscriber, while in the 2 Publishers case, one of the two Publishers is closer from the Subscriber. In the case of the 24 Publishers, which is a full load traffic, the delay slightly increases. It can be attributed to the increase in buffering and processing at the relay nodes for the packets that take the same path to the Subscriber. It should be noted the TDDS_AES_SHA1 EED follows the same behavior as the TDDS EED, but in all cases Publishers it takes longer time due to extra computations for security components.

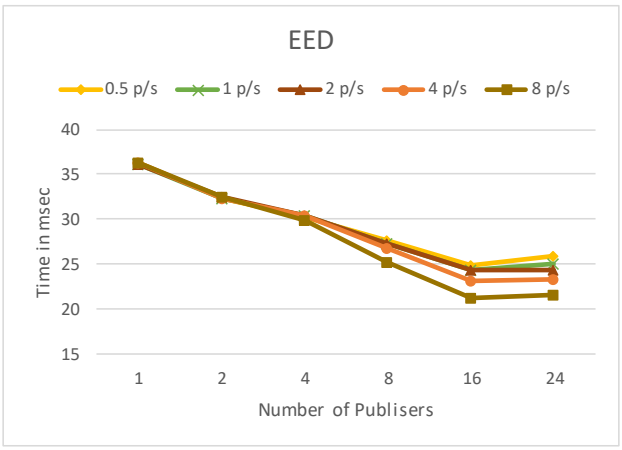
(a) TDDS vs TDDS_AES_SHA1 **(b) All three MW versions**

Figure 18: End-to-End delay comparison for all MW versions when the data rate is 1p/s.

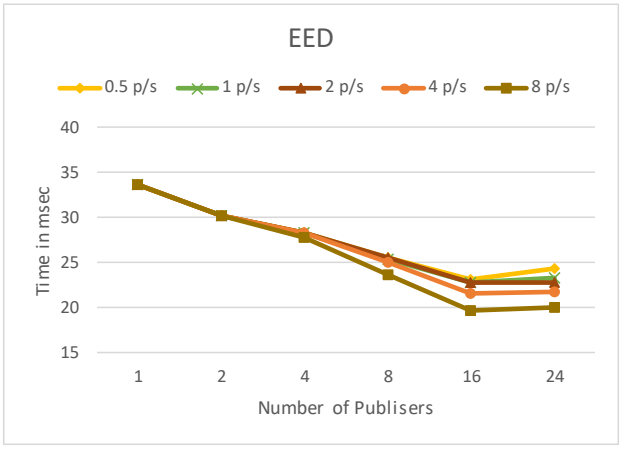
Figure 19 compares the EED results obtained from changing the data rate from 0.5 p/s to 8 p/s for all cases. It is apparent from the figure that all versions have similar plot trend. When the number of Publishers increases from 1 to 16, the value of the EED decreases for all data rates. Also, the value of EED is almost the same, but with a slight difference for data rate 8 p/s when the number of Publisher is 8 and 16, the value of EED is less than the other data rate. It can be attributed to the effect of the increase in the packet loss for the packets coming from the farther nodes when the data rate increases. However, when the number of Publishers is 24 (full load) the value of the EED increases as the network congestion affects the processing time at relay nodes of all packets that take the same path to the Subscriber. At full load, the network congestion effect may have more impact on the overall delay than the packet loss of the packets arriving from the remote nodes. It leads to increase the delay in the same data rate, but still lower when compared to lower data rates.



(a) Results for TDDS



(b) Results for TDDS_AES_SHA1



(c) Results for TDDS_AES_MMH

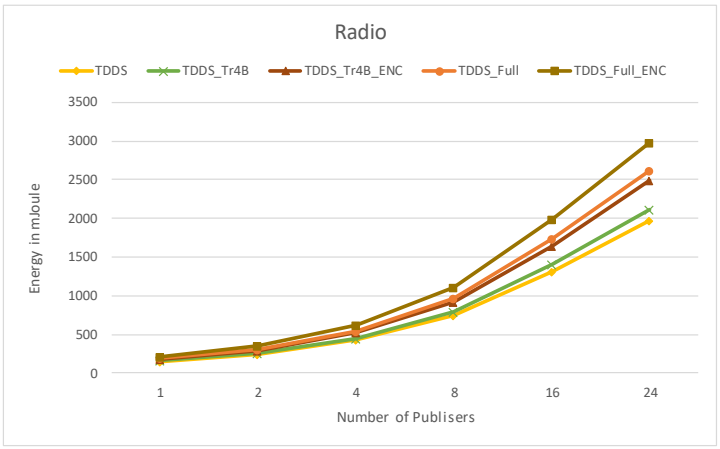
Figure 19: End-to-End delay results for all MW versions with different data rates.

5.3 Results of applying truncation for MAC in Integrity

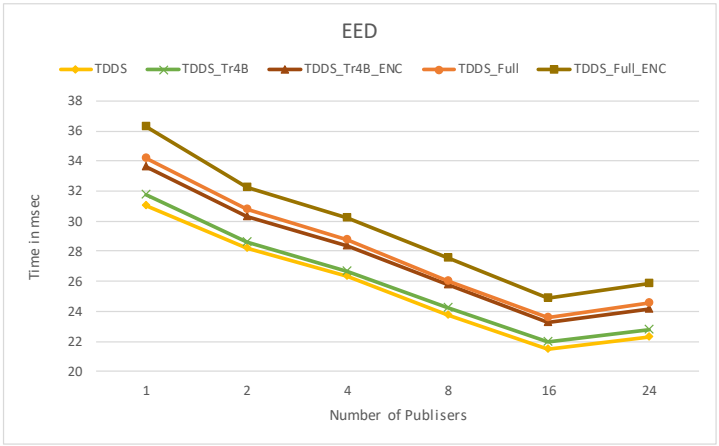
The MAC truncation is usually applied in WSNs to improve the performance network. The concern of this method is in losing the level of security due to increasing the probability of collisions as mentioned in subsection 3.2. Thus, it can be applied to produce 4 bytes MAC from 20 bytes HMAC-SHA1. The size of the input block in the standard SHA1 algorithm is 64 bytes, and the output block size is 20 bytes. We have applied some variations as explained next when using MAC truncation. The result of truncation is 4 bytes, and the variations are on whether to encrypt the MAC or leave it without encryption.

Figure 20 presents the results of applying the variations of truncation when the data rate is 0.5 p/s. Each line in the figure reflects the variations of applying MAC truncation. There are five versions of the middleware; TDDS, TDDS_Tr4B, TDDS_Tr4B_ENC, TDDS_FULL, and. TDDS_FULL_ENC. TDDS represents the basic TinyDDS which has by default 16 bytes of packet size. While the TDDS_Tr4B is the modified version of the middleware where the truncation is 4 bytes with no encryption resulting in a total packet size of 20 bytes. On the other hand, TDDS_Tr4B_ENC is the encrypted truncated 4 bytes middleware version, resulting in a total packet size of 32 bytes. TDDS_FULL uses the full 20 bytes MAC that is produced by the SHA1 resulting in a total packet size of 36 bytes. The last version is TDDS_FULL_ENC, which is the encrypted version of TDD_FULL resulting in a in 48 bytes packet. TDDS_FULL_ENC provides the highest level of security and it is the secure middleware using HMAC-SHA1 for integrity version that was presented in the figures presented in section 5.2. Figure 20 shows a considerable reduction in the energy consumption and the end-to-end delay results when comparing TDDS_Tr4B with

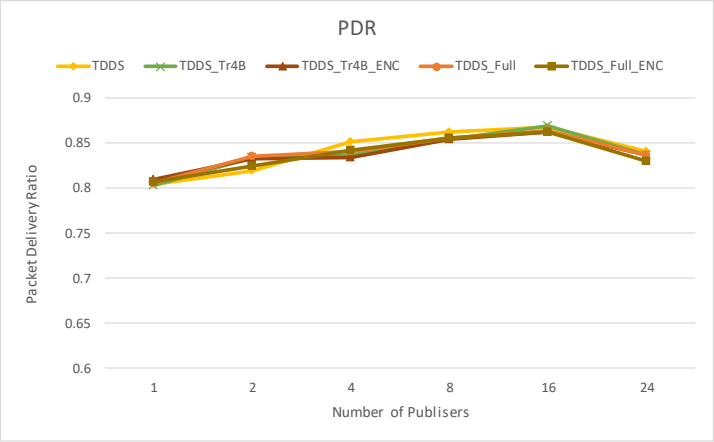
TDDS_FULL_ENC. The results for the later middleware version are almost equal to the basic TindyDDS results. The difference in the packet delivery ratio results for the different middleware versions is negligible since the data rate used is low. Therefore, the packet loss is minimized. The security level is not the same for the different middleware versions. Therefore, selecting which version to choose will have an impact on final the results.



(a) Energy consumption



(b) End-to-end delay comparison



(c) Packet delivery ratio comparison

Figure 20: Results of MAC truncation

5.4 Results of applying other lightweight block ciphers

In this section, we investigate using some of the recommended lightweight block cipher algorithms to study their effect in our implementation as mentioned in subsection 3.2. We compare the lightweight block ciphers results with the results of AES which was used to achieve confidentiality in our modified TinyDDS middleware. The same simulation parameters are applied for the network configuration setup. The lightweight block ciphers considered are, Speck-Simon, Present, and Prince. They are selected due to their advantage mainly in the security level and in the memory requirement. The minimum acceptable key size for our implementation is 128 bits. Table 13 provides an overview of the memory occupancy of the selected lightweight block ciphers as compared to the standard AES algorithm. Accordingly to Table 10, the block cipher algorithm that requires the least memory is the Speck-Simon algorithm. It requires less than half of the memory required by AES. Although these algorithms are lightweight block ciphers, the Prince algorithm

requires more ROM memory than the other algorithms while a very low amount of RAM memory.

Table 13: Memory Occupancy of lightweight block ciphers.

	ROM (Bytes)	RAM (Bytes)	ROM (%)	RAM (%)
TDDS	23324	5952	47.5	58.1
AES	4014	1810	8.2	17.7
Speck	1798	736	3.6	7.2
Present	3792	1664	7.7	16.2
Prince	6024	1112	12.3	10.9

Using the same simulation parameters, the performance of the network using the selected light block ciphers is evaluated. The size of the input block for encryption contributes directly to the packet size. The benefit of lightweight block cipher over AES is in the smaller block size requirement. That is, the resultant packet length is when using the lightweight block ciphers. The minimum block size of AES algorithm is 128 bits, whereas, the block size of Speck algorithm can be 32,48,64,96 or 128 bits. We found that using a block of size 32 bits can reduce the overall packet size to be just 36 bytes, whereas using AES algorithm, the size of the packet is 48 bytes when applying full MAC with encryption. The block size of Prince and Present is 64 bits. Similarly, the size of the resultant packet is 40 bytes. Since both algorithms have the same block size, we provide the results of only the Present algorithm, especially that it requires less memory than the Prince algorithm. The size of the original packet for the basic TinyDDS is 16 bytes. The data rate used in the performance evaluation of the results is 1 p/s.

Figure 21 shows the results after using the Speck lightweight block cipher algorithm. The Speck algorithm results follow the same graph trend of the TDDS results. Similarly, Figure

22 shows the results after using the Present lightweight block cipher algorithm. However, the results of the Speck algorithm and the Present algorithm are less than the results of the AES algorithm, which is presented when we combine all results on the same figure, especially, for the Radio consumption and end-to-end delay results.

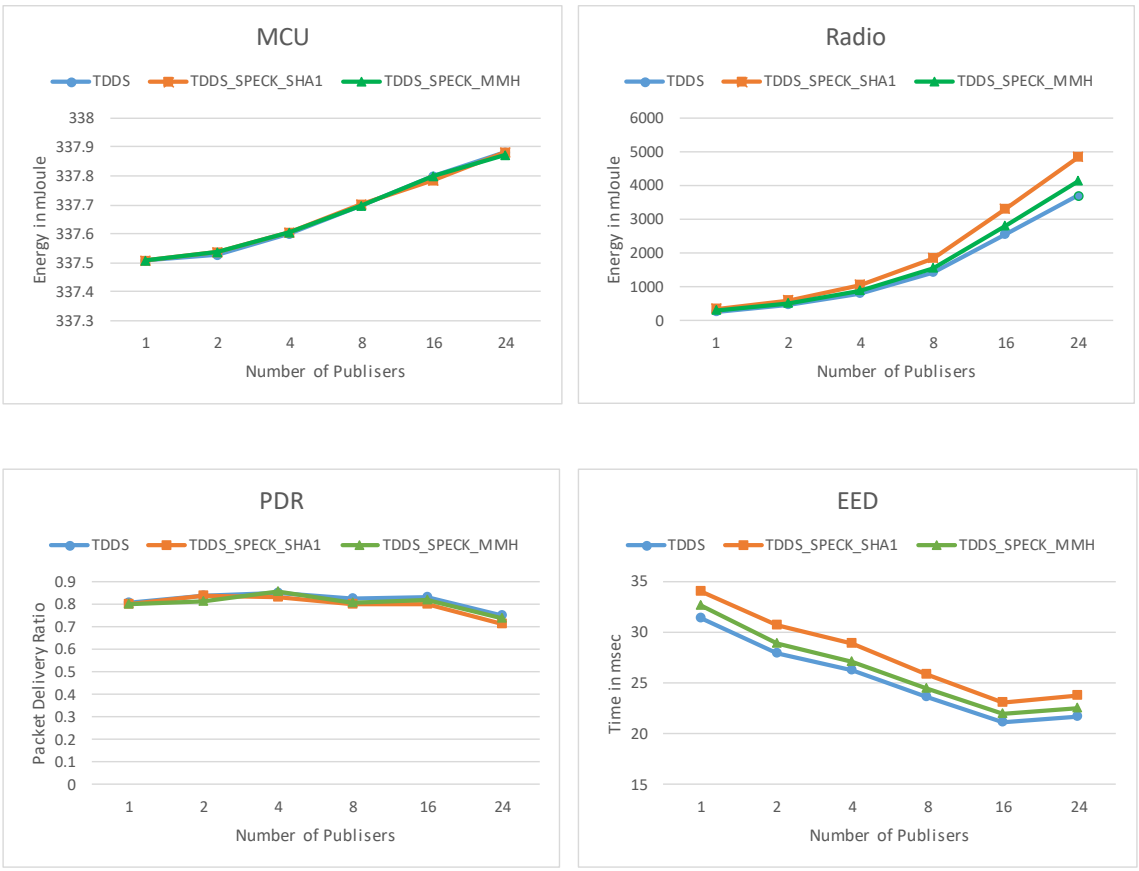


Figure 21: Results for using Speck lightweight block cipher.

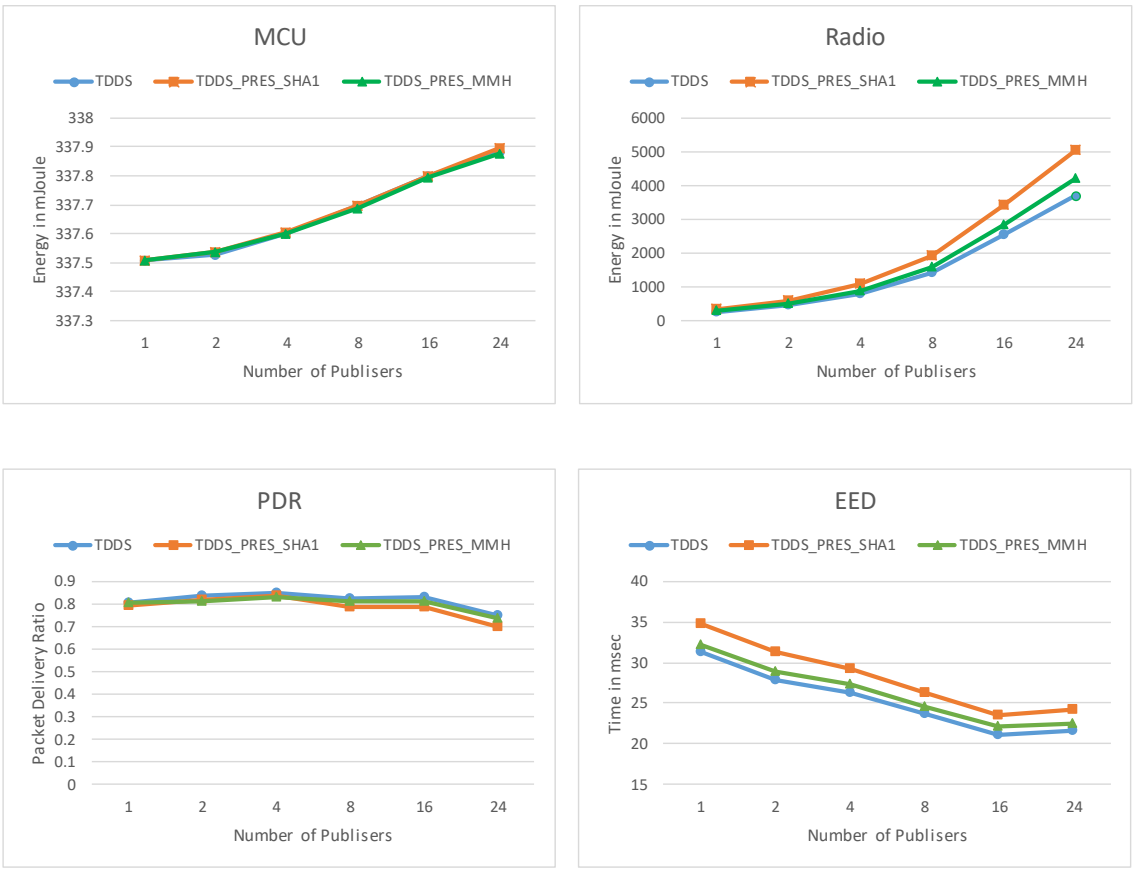
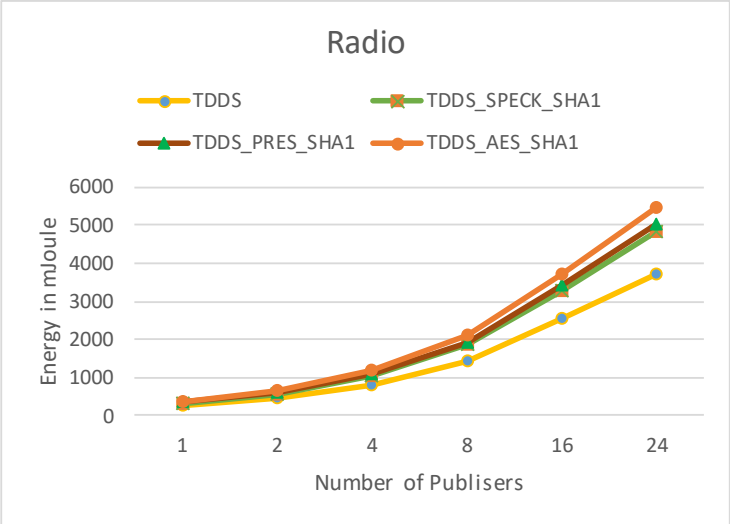
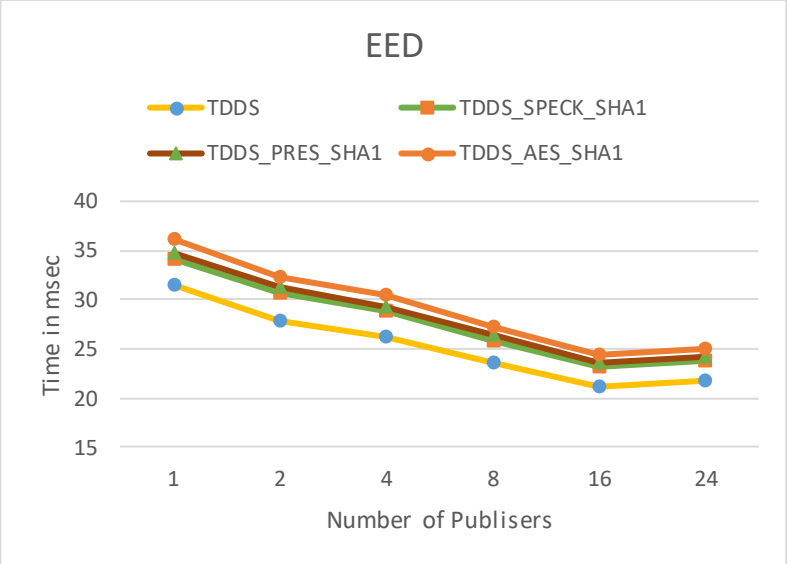


Figure 22: Results for using Present lightweight block cipher.

Figure 23 shows the results of the AES, Present, and Speck algorithms when the data rate is 1p/s. It can be seen that the results of AES are the highest while the results of the Speck algorithm are the lowest. This confirms the advantage of using a lightweight block ciphers over a standard block cipher.



(a) Radio energy consumption.



(b) End-to-end delay

Figure 23: Comparison of the three algorithms when data rate is 1p/s

5.5 Network size impact on performance

In this section, we investigate the impact of network size on the performance results. The performance results reported so far were for a network size of 5x5. Accordingly, we consider two more network sizes, 3x3 and 4x4, while maintaining the same distance between the nodes as in the case of 5x5 network size. Following the same notations as previous, Figures 24 and 25 show the distribution of the nodes and the location of the Base-station and the Publishers for network size 4x4 and 3x3, respectively.

Following the same simulation parameters used for the 5x5 network size, we produce the results for three confidentiality algorithms, namely, AES, Present and Speck, and two integrity algorithms, SHA1 and MMH. The data rate used to produce the results is 1p/s to reduce the overall simulation time, and the packet delivery ratio is almost constant when increasing the number of publishers. Figures 26-28 demonstrate the results for the network size of 4x4. Similarly, Figures 29-31 demonstrate the results for the network size of 3x3.

The results of network size 4x4 and 3x3 are following the same general trend for the results of network size 5x5. As in contributing to the extra overhead of the modified middleware compared with the basic middleware. The results for each confidentiality algorithm is produced separately and plot in the same figure for both integrity algorithms, SHA1 and MMH. The results of supporting integrity with SHA1 are contributing on more overhead than when supporting integrity with MMH algorithm, as observed from the results of the network size of 5x5.

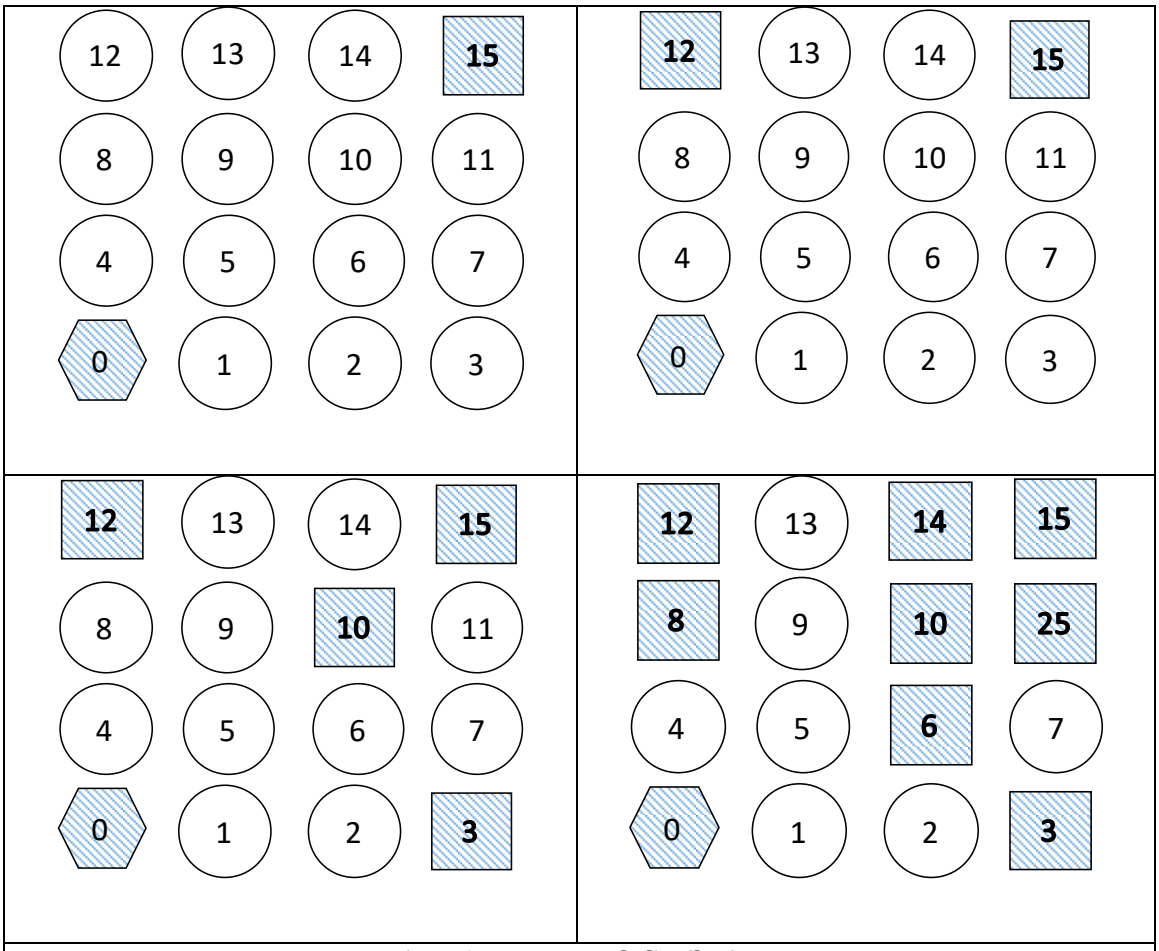


Figure 24: 4x4 Network distribution.

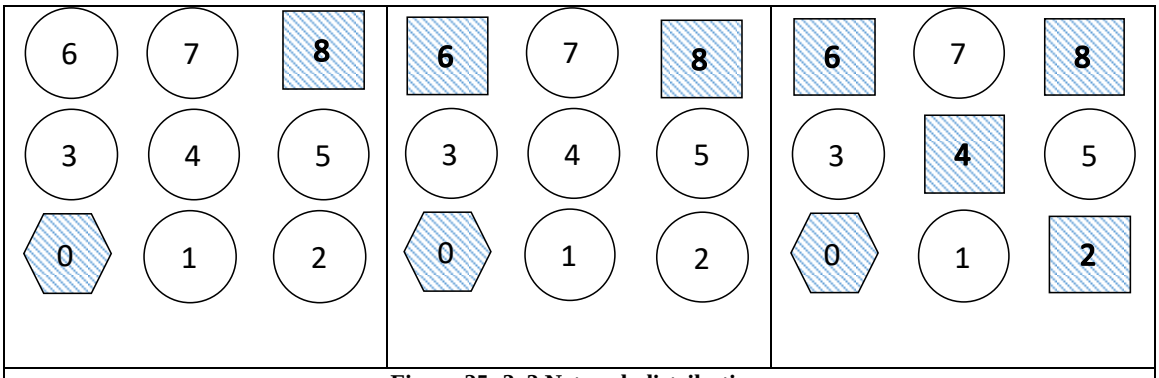


Figure 25: 3x3 Network distribution.

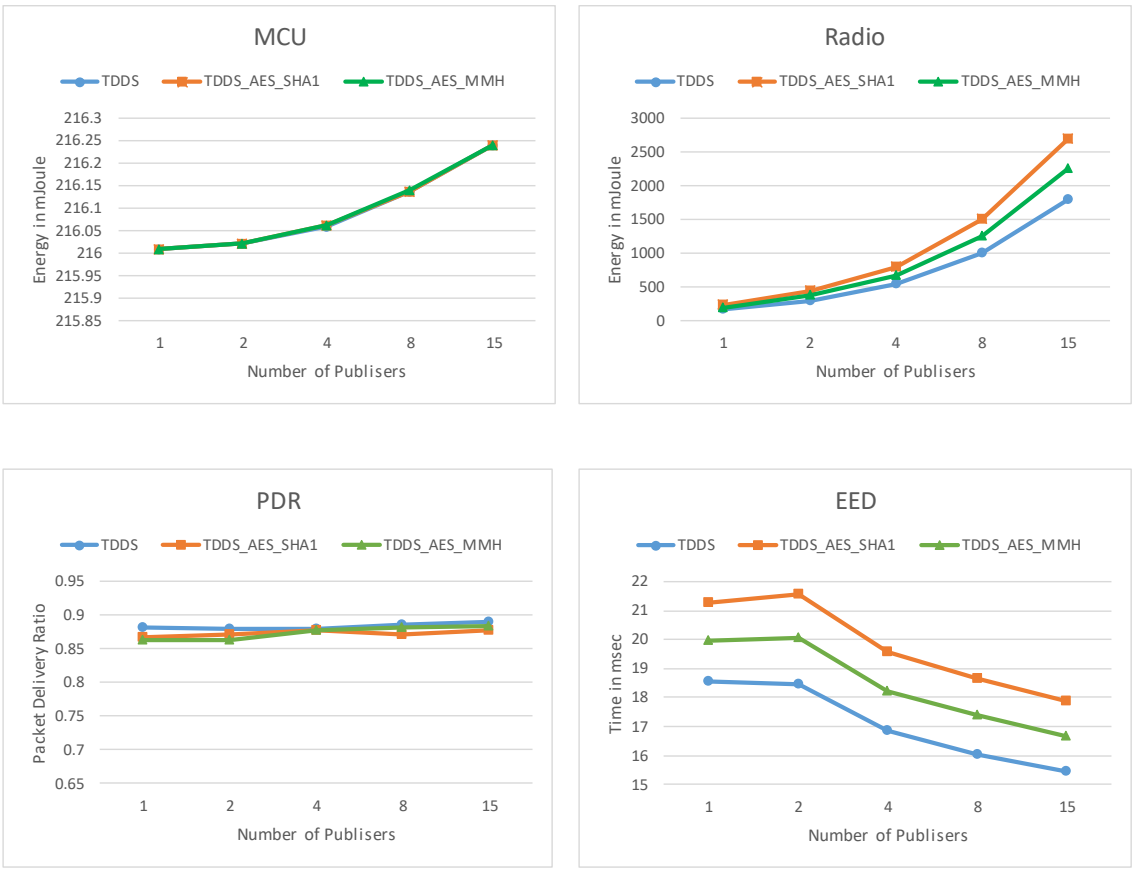


Figure 26: AES results for 4x4 network.

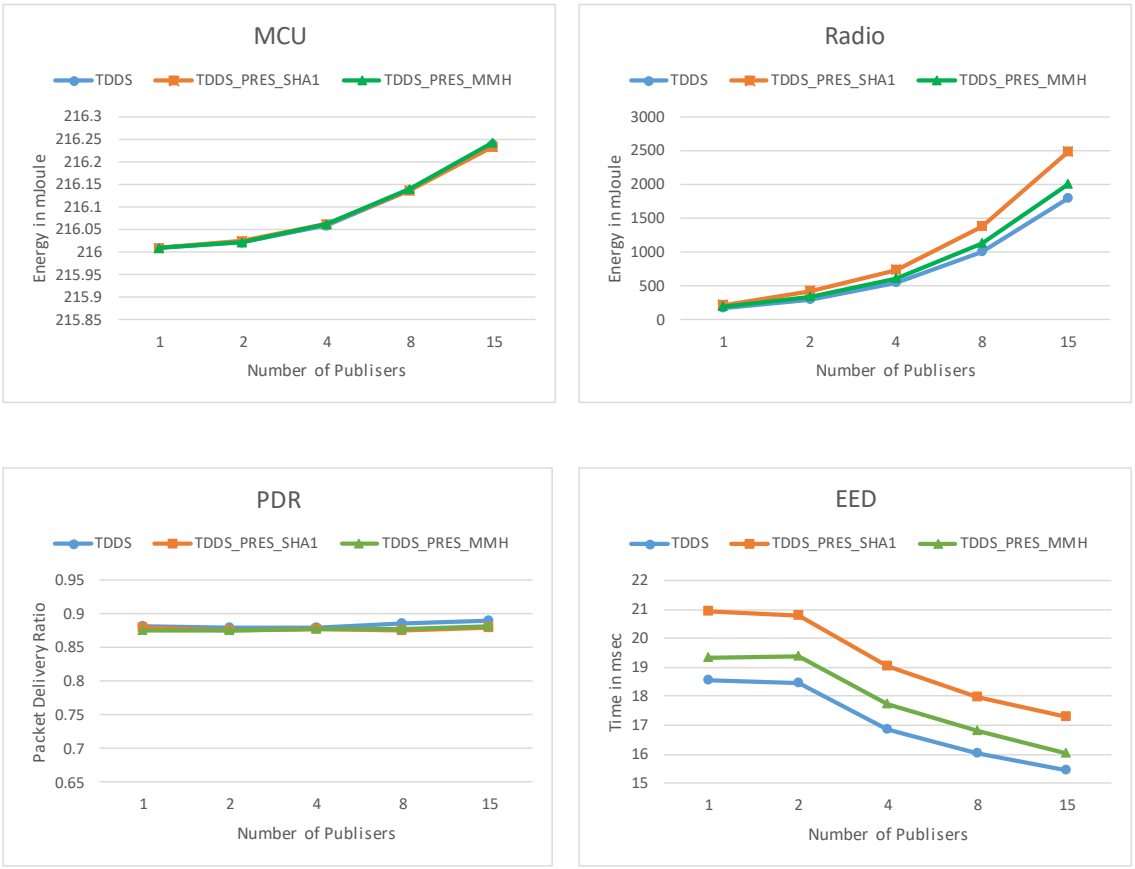


Figure 27: Present results for 4x4 network

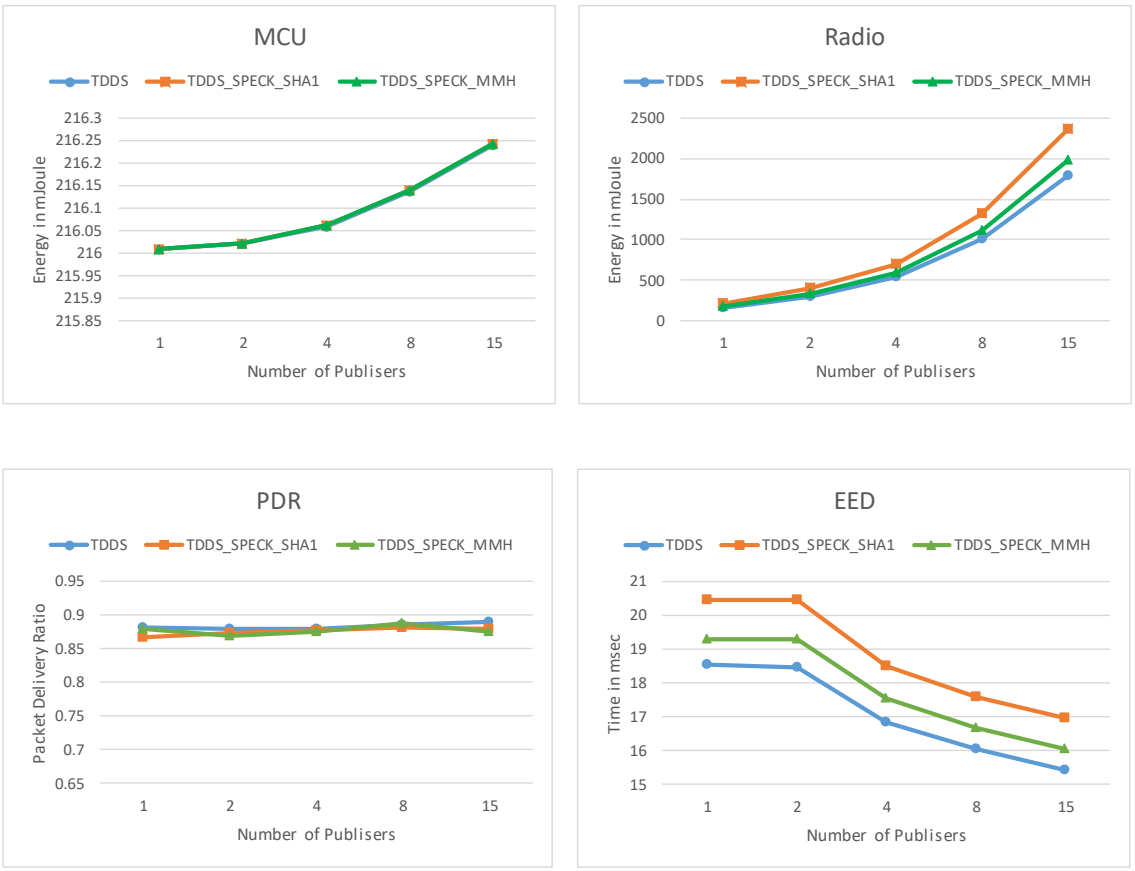


Figure 28: Speck results for 4x4 network.

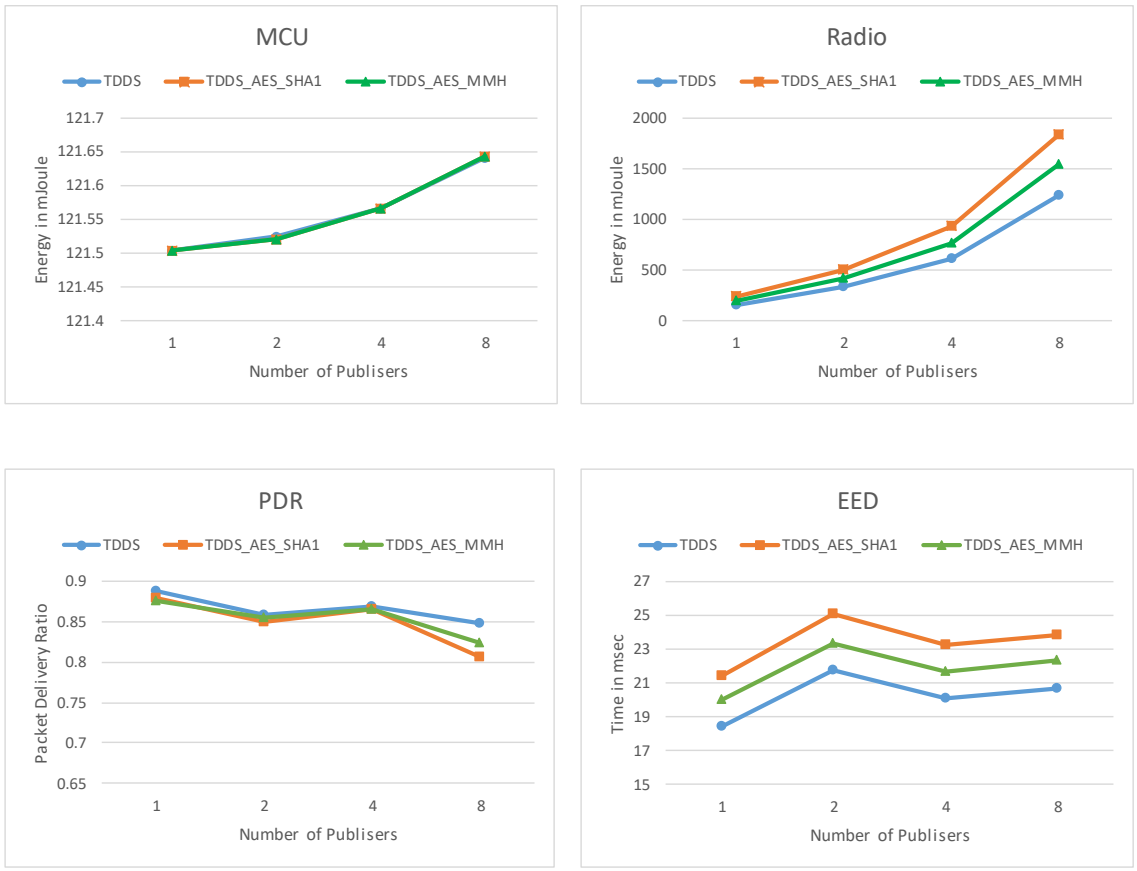


Figure 29: AES results for 3x3 network.

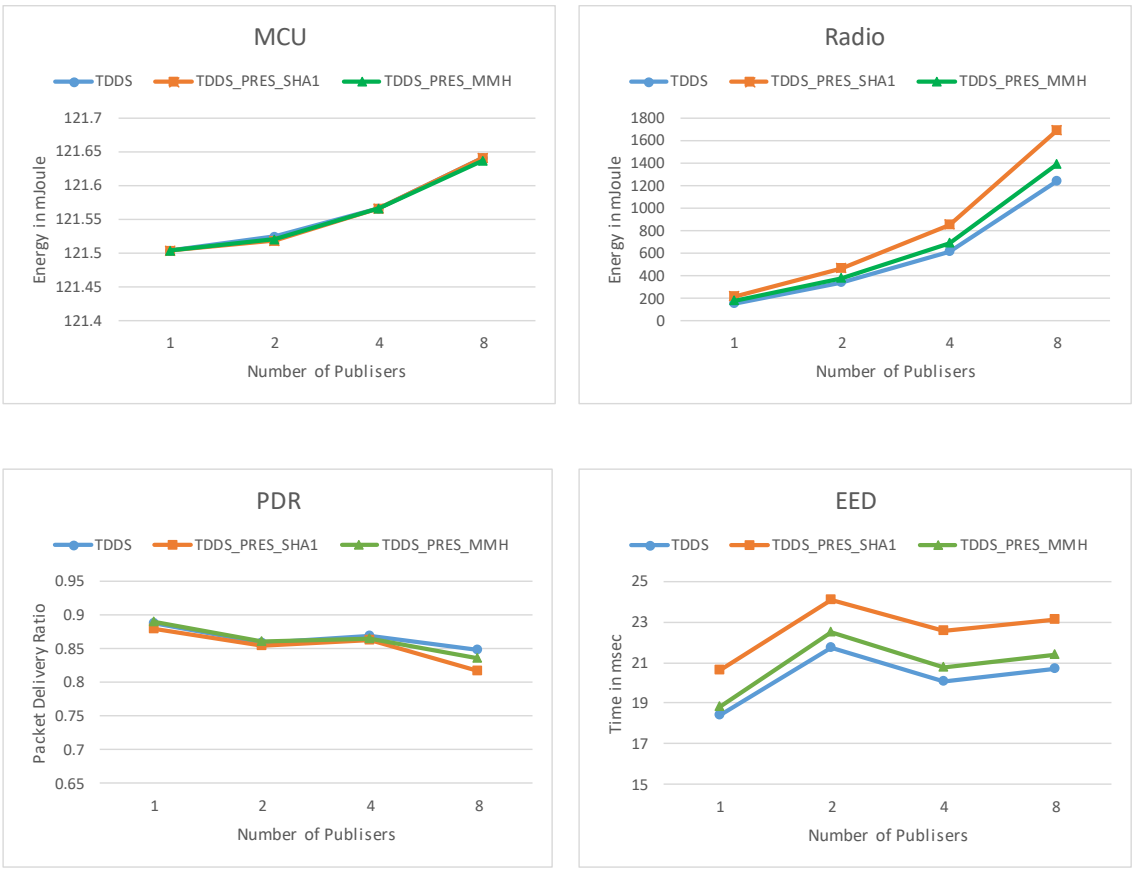


Figure 30: Present results for 3x3 network.

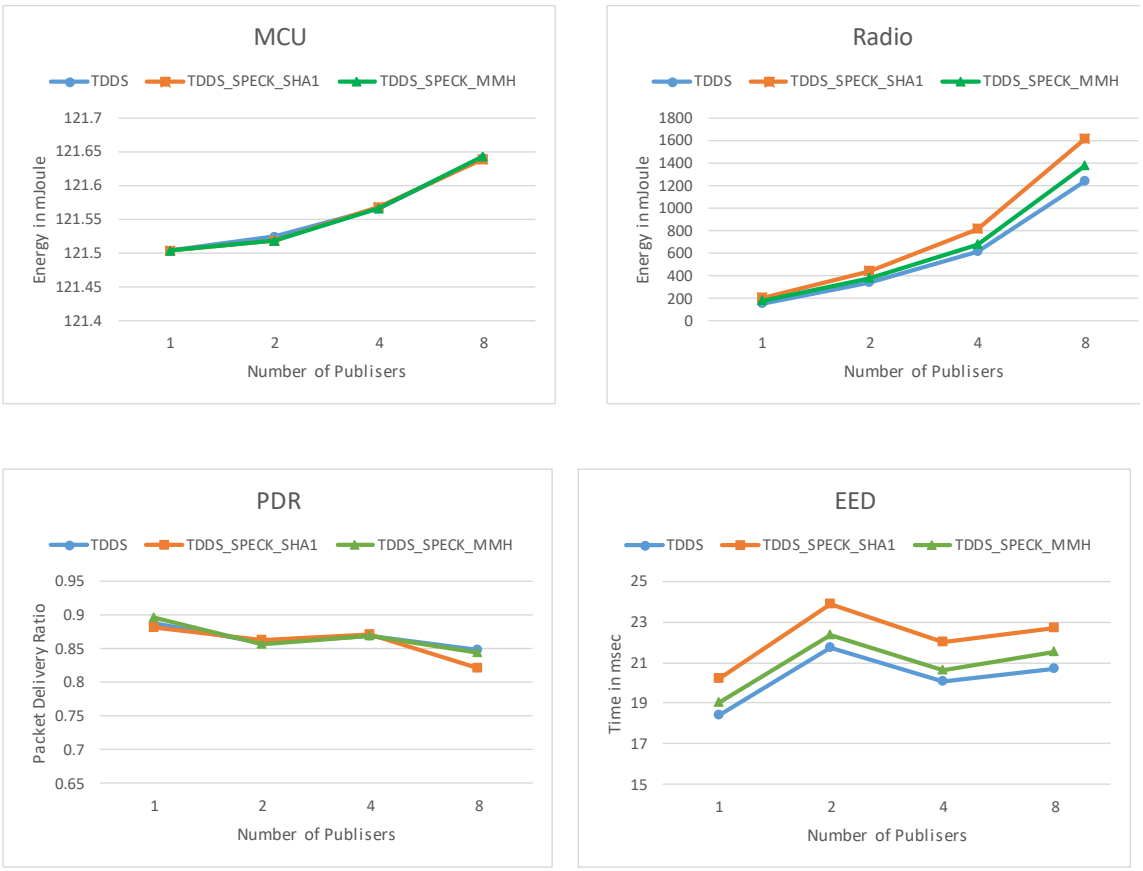


Figure 31: Speck results for 3x3 network.

Scaling the size of the network maintains the same results trend when comparing them regarding of the two integrity algorithms for each of the confidentiality algorithms. However, the observation in the EED results for the network size of 3x3 in Figures 29-31 is not similar when comparing them with the network size of 5x5. The delay does not decrease as the number of Publishers increases from 1 to 8 as expected and observed from the network size of 5x5. On the contrary, the value of EED is the lowest when we have only one Publisher in the network, which is observed for all confidentiality algorithms. For the network size of 4x4, the value of EED is roughly the same when the number of Publishers is 1 and 2, but similar to the network size of 5x5, the value of EED decreases as the number of Publishers increases.

5.6 Authentication and key exchange support cost

The authentication and key exchange process must be completed before exchanging data messages in the network. The key exchange is needed when the validity of the key expires. The frequency of initiating the key exchange protocol depends on the duration of the session key expiration time, and impacts the overall energy cost. To estimate the cost of one operation of authentication and key exchange, we assumed two scenarios, best-case and worst case. The best-case scenario happens when the publisher node is a direct neighbor to the TTP node, whereas the worst-case scenario happens when the publisher node is at the far end of the 5x5 network, node ID 24. We estimate the energy consumption of the network as in previous experiments. The authentication and key exchange messages are sent repeatedly for different number of times, then the average for both cases is taken. The number of repetitions used are 100, 150, 200, and 400. The average value of the energy consumption of completing one authentication and key exchange messages is 2.17 mJ for the best-case scenario, and 4.36 mJ for the worst-case scenario. The memory requirements for supporting the protocol in the modified middleware is 656 bytes for ROM and 182 bytes for RAM, which adds 1.3% and 1.8% on contribution of the modified middleware on Table 11 in subsection 5.2.1 for ROM and RAM, respectively.

CHAPTER 6

EXTENDED ANALYSIS AND DISCUSSION

In this chapter, we extend the simulation results for additional scenarios. Availability support is discussed to be included in the modified version of the middleware. We propose a three-broker case for the middleware to reduce the congestion of the packets at a single broker solution. The method can be extended to be as cluster-based brokers distributed for networks with a large number of nodes. A mathematical approximation model is discussed to estimate the energy consumption of the radio transmission for the nodes. Also, a space complexity model is investigated to compare the results of the algorithm we used to support the modified middleware. Packet loss analysis is performed to study the readings of the individual nodes and their direct contribution on the overall values observed.

6.1 Detail simulation results

In this section, more detail regarding the simulation results and the complete routing paths dedicated to every publisher in the network. One of the major strengths of The TinyDDS middleware is to allow us to change the implementation of any layer of its architecture. Our contribution is mainly to add the security service to the middleware architecture. To investigate and study the system, we can track all messages sent from any node along routing paths until they reach the base station. Therefore, the Id-based [86] routing algorithm is used for the network routing in the simulation. It is designed for grid topology network distribution. It ensures the packets to take the shortest path. In which, the route

moves diagonally until it reaches the destination row or column, and then it moves horizontally or vertically respectively, to reach the destination node. Figure 32 shows the distribution the Publishers in the network for all the cases. Table 16 shows the detail routing paths, and the number hops needed to reach the destination, for all the nodes according to the Id-based routing algorithm. The TinyDDS in our implementation is a broker-based. The broker node acts as a central node, to forward the messages from the publishers to the subscribers. This method is to support the decoupling properties of the middleware. The broker node is set to node 6 in our simulation results, to allow us to track the messages along the routing paths. We collected the simulation results for publishing rate one msg/s for all versions of the secured middleware and compared it with the basic implementation. The reliability of delivering packets is set to be the best effort. The drop in the packet is due to signal interference between the nodes. The capacity of the transferring the messages over the radio will be explained in a separate section later. The following discussion is for the results of basic TinyDDS implementation that are shown in Figure 33. In the results, there is always a drop in the number of received messages that can reach 5% of the total sent messages. The parameter is set to best effort; there is no retransmission of the lost messages. Later on, we investigate improving the radio and channel parameters, so that we eliminate that the lost messages are due to the radio connectivity. Decreasing the distance between the nodes and increasing their gain power leads to reducing the number of dropped packets. It is applied by giving the TOSSIM simulator the topology in a file, which creates realistic values. With a Python script, we can load values and store them into the radio object. For example, the Python script reads the file which specifies each link in between the nodes as a line with three values, the source, the destination, and the gain. The line in

the file as the following: “1 2 -54.0”, means that when node ID_1 transmits, node ID_2 hears it at -54 dBm.

Table 14 shows the parameters to specify the topology of our network to generate the values for the nodes in the first column, and a description of the parameters, in the second columns. Table 15 shows part of the values from the file that is generated from the parameters of the network topology.

Table 14: Simulator tool parameters for the channel and the radio

PATH_LOSS_EXPONENT = 4.7; SHADOWING_STANDARD_DEVIAT ION = 3.2; D0 = 1.0; PL_D0 = 55.4; NOISE_FLOOR = -105.0; S11 = 0.9; S12 = -0.7; S21 = -0.7; S22 = 1.2; WHITE_GAUSSIAN_NOISE = 4; NUMBER_OF_NODES =25;	**Channel Parameters:** PATH_LOSS_EXPONENT: the rate at which signal decays. SHADOWING_STANDARD_DEVIAT ION: the randomness of the received signal due to multipath. D0: reference distance (usually 1 meter). D0 also determines the minimum distance allowed between any pair of nodes. PL_D0: power decay in dB for the reference distance D0. **Radio Parameters:** WHITE_GAUSSIAN_NOISE: standard deviation of the additive white Gaussian noise. NOISE_FLOOR: radio noise floor in dBm. Moreover, the variances of the output power and noise floor on a per-node basis are given by the covariance matrix S = [S11 S12; S21 S22]: S11: variance of the noise floor. S12: covariance between the noise floor and output power (captures correlation). S21: equal to S12. S22: variance of output power.

Table 15: Network nodes values.

gain 0	1	-58.15	gain 0	9	-81.61
gain 1	0	-57.25	gain 9	0	-83.00
gain 0	2	-64.90	gain 0	10	-65.92
gain 2	0	-65.61	gain 10	0	-66.43
gain 0	3	-78.68	gain 0	11	-74.39
gain 3	0	-77.26	gain 11	0	-74.51
gain 0	4	-84.08	gain 0	12	-70.18
gain 4	0	-85.13	gain 12	0	-72.33
gain 0	5	-53.24	gain 0	13	-74.85
gain 5	0	-53.94	gain 13	0	-73.56
gain 0	6	-62.79	gain 0	14	-80.02
gain 6	0	-61.84	gain 14	0	-81.01
gain 0	7	-74.15	gain 0	15	-74.72
gain 7	0	-75.61	gain 15	0	-72.57
gain 0	8	-79.58	gain 0	16	-80.77
gain 8	0	-79.82	gain 16	0	-81.32

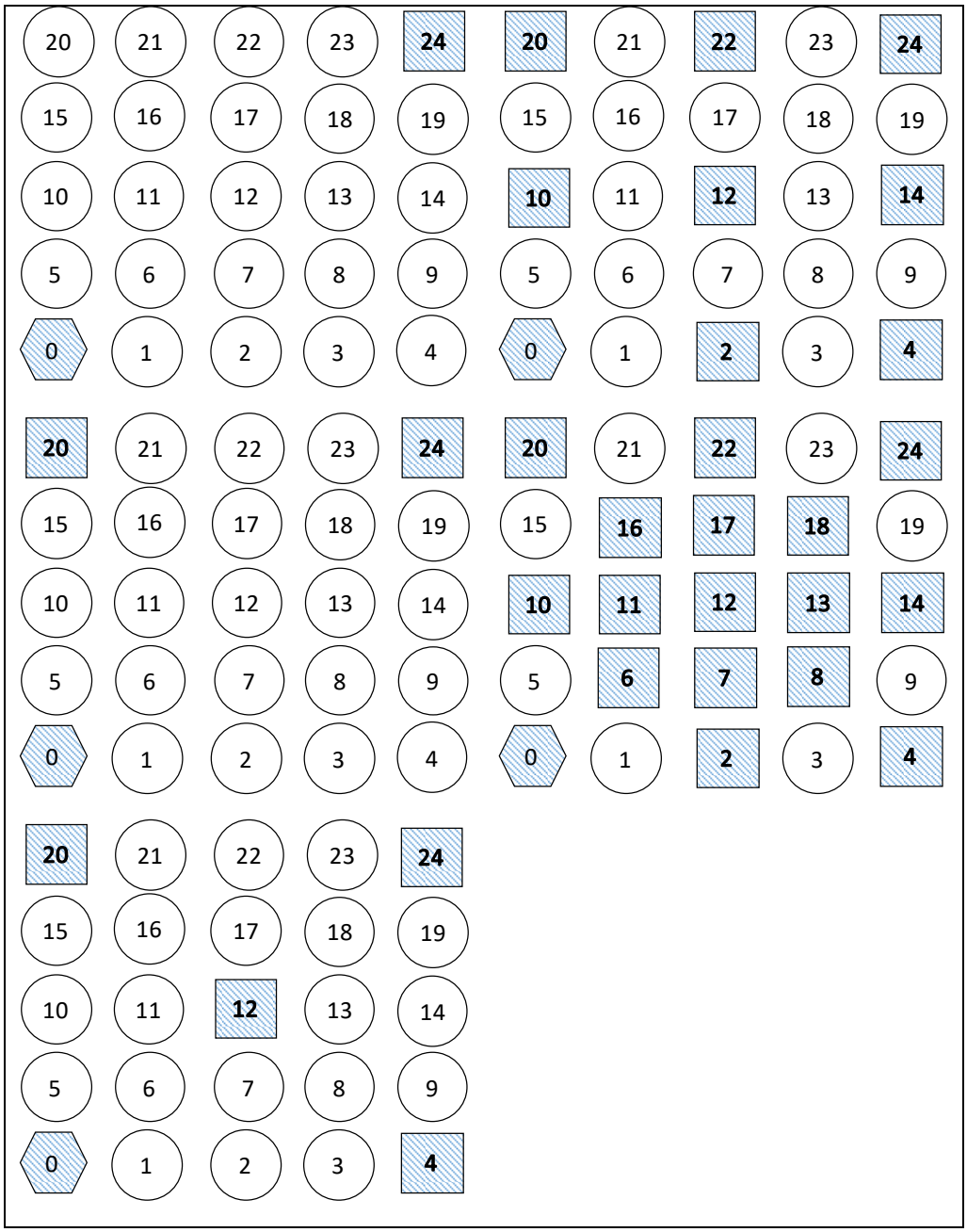


Figure 32: 5x5 Publishers distribution.

Table 16: Routing detail for all nodes.

Publisher-ID:1 1-6-0 Hops=2	Publisher-ID:13 13-7-6-0 Hops=3
Publisher-ID:2 2-6-0 Hops=2	Publisher-ID:14 14-8-7-6-0 Hops=4
Publisher-ID:3 3-7-6-0 Hops=3	Publisher-ID:15 15-11-6-0 Hops=3
Publisher-ID:4 4-8-7-6-0 Hops=4	Publisher-ID:16 16-11-6-0 Hops=3
Publisher-ID:5 5-6-0 Hops=2	Publisher-ID:17 17-11-6-0 Hops=3
Publisher-ID:6 6-5-6-0 Hops=3	Publisher-ID:18 18-12-6-0 Hops=3
Publisher-ID:7 7-6-0 Hops=2	Publisher-ID:19 19-13-7-6-0 Hops=4
Publisher-ID:8 8-7-6-0 Hops=3	Publisher-ID:20 20-16-11-6-0 Hops=4
Publisher-ID:9 9-8-7-6-0 Hops=4	Publisher-ID:21 21-16-11-6-0 Hops=4
Publisher-ID:10 10-6-0 Hops=2	Publisher-ID:22 22-16-11-6-0 Hops=4
Publisher-ID:11 11-6-0 Hops=2	Publisher-ID:23 23-17-11-6-0 Hops=4
Publisher-ID:12 12-6-0 Hops=2	Publisher-ID:24 24-18-12-6-0 Hops=4

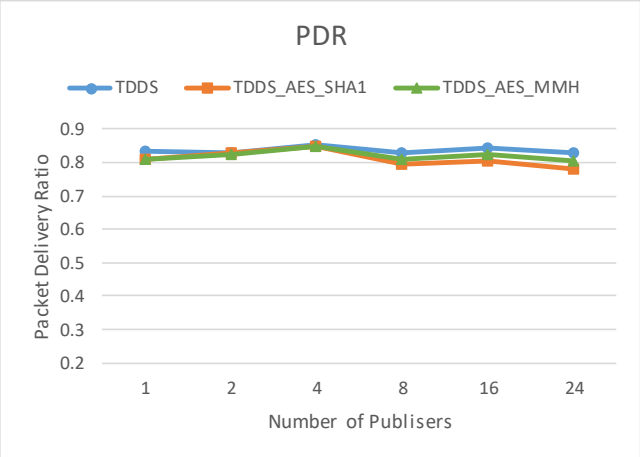


Figure 33: PDR results for the 5x5 network.

The PDR details of the individual publishers for all cases of some publishers, and the publishing rate is one msg/s, are as the following:

One Publisher node 24:

Publisher 24:

The routing path for the packet is: 24-18-12-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (24) is 500. The number of messages received by Subscriber (0) is 418. The PDR is 0.836.

Details in the relay nodes in between the path 24-18-12-6-0:

(24): sent: **500** to (18) Received: 482, sent to (12): 482.

(12) Received: 454, sent to (6): 454. (6) Received: 437, sent to (0) 437.

Received: **418**.

Two Publishers node 24 and 20:

Publisher 24:

The routing path for the packet is: 24-18-12-6-0

The number of messages sent by Publish (24) is 500. The number of messages received by Subscriber (0) 405 PDR 0.81

Details in the relay nodes in between the path 24-18-12-6-0:

(24): sent: **500** to (18) Received: 471, sent to (12): 471.

(12) Received: 447, sent to (6): 447. (6) Received: 434, sent to (0) 434.

Received: **405**.

Publisher 20:

The routing path for the packet is: 20-16-11-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (20) is 500. The number of messages received by Subscriber (0) is 412 PDR 0.824.

Details in the relay nodes in between the path 20-16-11-6-0:

(20): sent: **500** to (16) Received: 488, sent to (11): 488.

(11) Received: 461, sent to (6): 461. (6) Received: 444, sent to (0) 444.

Received: **412**.

Four Publishers node 24, 20, 12, 4:

Publisher 24:

The routing path for the packet is: 24-18-12-6-0

The number of messages sent by Publish (24) is 500. The number of messages received by Subscriber (0) 410 PDR 0.82

Details in the relay nodes in between the path 24-18-12-6-0:

(24): sent: **500** to (18) Received: 473, sent to (12): 473.

(12) Received: 452, sent to (6): 452. (6) Received: 433, sent to (0) 433.

Received: **410**.

Publisher 20:

The routing path for the packet is: 20-16-11-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (20) is 500. The number of messages received by Subscriber (0) is 412 PDR 0.824.

Details in the relay nodes in between the path 20-16-11-6-0:

(20): sent: **500** to (16) Received: 477, sent to (11): 477.

(11) Received: 465, sent to (6): 465. (6) Received: 456, sent to (0) 456.

Received: **435**.

Publisher 4:

The routing path for the packet is: 4-8-7-6-0

The number of messages sent by Publish (4) is 500. The number of messages received by Subscriber (0) 411 PDR 0.822

Details in the relay nodes in between the path 4-8-7-6-0 :

(4): sent: **500** to (8) Received: 471, sent to (7): 471.

(7) Received: 447, sent to (6): 447. (6) Received: 432, sent to (0) 432.

Received: **411**.

Publisher 12:

The routing path for the packet is: 12-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (20) is 500. The number of messages received by Subscriber (0) 458 PDR 0.916

Details in the relay nodes in between the path 12-6-0 :

(12): sent: **500** to (6) Received: 479, sent to (0): 479.

Received: **458**.

Eight Publishers: node 24,20,12,4,2,10,14,22:

Publisher 24:

The routing path for the packet is: 24-18-12-6-0

The number of messages sent by Publish (24) is 500. The number of messages received by Subscriber 405 PDR 0.81

Details in the relay nodes in between the path 24-18-12-6-0:

(24): sent: **500** to (18) Received: 474, sent to (12): 474.

(12) Received: 453, sent to (6): 453. (6) Received: 435, sent to (0) 435.

Received: **405**.

Publisher 20:

The routing path for the packet is: 20-16-11-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (20) is 500. The number of messages received by Subscriber (0) is 397 PDR 0.794

Details in the relay nodes in between the path 20-16-11-6-0:

(20): sent: **500** to (16) Received: 447, sent to (11): 447.

(11) Received: 432, sent to (6): 432. (6) Received: 419, sent to (0) 419.

Received: **397**.

Publisher 4:

The routing path for the packet is : 4-8-7-6-0

The number of messages sent by Publish (4) is 500. The number of messages received by Subscriber (0) 427 PDR 0.854

Details in the relay nodes in between the path 4-8-7-6-0 :

(4): sent: **500** to (8) Received: 474, sent to (7): 474.

(7) Received: 463, sent to (6): 463. (6) Received: 448, sent to (0) 448.

Received: **427**.

Publisher 12:

The routing path for the packet is: 12-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (12) is 500. The number of messages received by Subscriber (0) 437 PDR 0.874

Details in the relay nodes in between the path 12-6-0 :

(12): sent: **500** to (6) Received: 470, sent to (0): 470.

Received: **437**.

Publisher 22:

The routing path for the packet is: 22-16-11-6-0

The number of messages sent by Publish (22) is 499. The number of messages received by Subscriber 338 PDR 0.67735

Details in the relay nodes in between the path 22-16-11-6-0:

(22): sent: **499** to (16) Received: 483, sent to (11): 483.

(11) Received: 459, sent to (6): 459. (6) Received: 445, sent to (0) 445.

Received: **338**.

Publisher 14:

The routing path for the packet is: 14-8-7-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (14) is 500. The number of messages received by Subscriber (0) 399 PDR 0.79959

Details in the relay nodes in between the path 14-8-7-6-0 :

(14): sent: **499** to (8) Received: 471, sent to (7): 471.

(7) Received: 452, sent to (6): 452. (6) Received: 436, sent to (0) 436.

Received: **399**.

Publisher 2:

The routing path for the packet is: 2-6-0

The number of messages sent by Publish (2) is 500. The number of messages received by Subscriber (0) 450 PDR 0.9

Details in the relay nodes in between the path 2-6-0:

(2): sent: **500** to (6) Received: 476, sent to (0): 476.

Received: **450**.

Publisher 10:

The routing path for the packet is: 10-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (20) is 500. The number of messages received by Subscriber (0) 458 PDR 0.916

Details in the relay nodes in between the path 10-6-0 :

(10): sent: **500** to (6) Received: 483, sent to (0): 483.

Received: **458**.

Sixteen Publishers: node 24,20,12,4,2,10,14,22,6,7,8,11,13,16,17,18:

Publisher 24:

The routing path for the packet is : 24-18-12-6-0

The number of messages sent by Publish (24) is 500. The number of messages received by Subscriber 403 PDR 0.806

Details in the relay nodes in between the path 24-18-12-6-0:

(24): sent: **500** to (18) Received: 471, sent to (12): 471.

(12) Received: 447, sent to (6): 447. (6) Received: 425, sent to (0) 425.

Received: **403**.

Publisher 20:

The routing path for the packet is: 20-16-11-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (20) is 500. The number of messages received by Subscriber (0) 391 PDR 0.782

Details in the relay nodes in between the path 20-16-11-6-0:

(20): sent: **500** to (16) Received: 452, sent to (11): 452.

(11) Received: 431, sent to (6): 431. (6) Received: 413, sent to (0) 413.

Received: **391**.

Publisher 4:

The routing path for the packet is: 4-8-7-6-0

The number of messages sent by Publish (4) is 500. The number of messages received by Subscriber (0) 420 PDR 0.84

Details in the relay nodes in between the path 4-8-7-6-0 :

(4): sent: **500** to (8) Received: 475, sent to (7): 475.

(7) Received: 452, sent to (6): 452. (6) Received: 436, sent to (0) 436.

Received: **420**.

Publisher 12:

The routing path for the packet is: 12-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (12) is 500. The number of messages received by Subscriber (0) 440 PDR 0.88

Details in the relay nodes in between the path 12-6-0 :

(12): sent: **500** to (6) Received: 470, sent to (0): 470.

Received: **440**.

Publisher 22:

The routing path for the packet is: 22-16-11-6-0

The number of messages sent by Publish (22) is 499. The number of messages received by Subscriber 341 PDR 0.683367

Details in the relay nodes in between the path 22-16-11-6-0:

(22): sent: **499** to (16) Received: 468, sent to (11): 468.

(11) Received: 443, sent to (6): 443. (6) Received: 424, sent to (0) 424.

Received: **341**.

Publisher 14:

The routing path for the packet is: 14-8-7-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (14) is 500. The number of messages received by Subscriber (0) 391 PDR 0.783567

Details in the relay nodes in between the path 14-8-7-6-0 :

(14): sent: **499** to (8) Received: 469, sent to (7): 469.

(7) Received: 448, sent to (6): 448. (6) Received: 431, sent to (0) 431.

Received: **391**.

Publisher 2:

The routing path for the packet is: 2-6-0

The number of messages sent by Publish (2) is 500. The number of messages received by Subscriber (0) 457 PDR 0.914

Details in the relay nodes in between the path 2-6-0:

(2): sent: **500** to (6) Received: 481, sent to (0): 481.

Received: **457**.

Publisher 10:

The routing path for the packet is: 10-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (20) is 500. The number of messages received by Subscriber (0) 453 PDR 0.906

Details in the relay nodes in between the path 10-6-0 :

(10): sent: **500** to (6) Received: 477, sent to (0): 477.

Received: **453**.

Publisher 6:

The routing path for the packet is: 6-5-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (6) is 499. The number of messages received by Subscriber (0) 449 PDR 0.8998

Details in the relay nodes in between the path 6-5-6-0 :

(6): sent: **499** to (5) Received: 483, sent to (6): 483.

Received: 469, sent to (0): 469. (6) Received: 431, sent to (0) 431.

Received: **449**.

Publisher 7:

The routing path for the packet is: 7-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (6) is 499. The number of messages received by Subscriber (0) 465 PDR 0.931864

Details in the relay nodes in between the path 7-6-0 :

(7): sent: **499** to (6) Received: 478, sent to (0): 478.

Received: **465**.

Publisher 8:

The routing path for the packet is: 8-7-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (8) is 499. The number of messages received by Subscriber (0) 424 PDR 0.849699

Details in the relay nodes in between the path 8-7-6-0 :

(8): sent: **499** to (7) Received: 472, sent to (6): 472.

Received: 457, sent to (0): 457. (0) Received: **424**.

Publisher 11:

The routing path for the packet is: 11-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (11) is 498. The number of messages received by Subscriber (0) 447 PDR 0.89759

Details in the relay nodes in between the path 11-6-0 :

(11): sent: **498** to (6) Received: 474, sent to (0): 474.

Received: **447**.

Publisher 13:

The routing path for the packet is: 13-7-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (13) is 498. The number of messages received by Subscriber (0) 422 PDR 0.84739

Details in the relay nodes in between the path 13-7-6-0 :

(13): sent: **498** to (7) Received: 463, sent to (6): 463.

Received: 442, sent to (0): 442. (0) Received: **422**.

Publisher 16:

The routing path for the packet is: 16-11-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (16) is 498. The number of messages received by Subscriber (0) 421 PDR 0.845382

Details in the relay nodes in between the path 16-11-6-0 :

(16): sent: **498** to (11) Received: 480, sent to (6): 480.

Received: 453, sent to (0): 453. (0) Received: **421**.

Publisher 17:

The routing path for the packet is: 17-11-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (17) is 498. The number of messages received by Subscriber (0) 372 PDR 0.746988

Details in the relay nodes in between the path 17-11-6-0 :

(17): sent: **498** to (11) Received: 483, sent to (6): 483.

Received: 465, sent to (0): 465. (0) Received: **372**.

Publisher 18:

The routing path for the packet is: 18-12-6-0

Publishing rate is one message per second. The simulation time is 500 seconds.

The number of messages sent by Publish (18) is 498. The number of messages received by Subscriber (0) 428 PDR 0.859438

Details in the relay nodes in between the path 18-12-6-0 :

(18): sent: **498** to (12) Received: 472, sent to (6): 472.

Received: 455, sent to (0): 455. (0) Received: **428**.

We have changed the radio parameters for the simulator to reducing the effect of radio interference by improving the transmitting and receiving power between nodes. The gain is increased between the nodes. Table 17 shows the new values of the nodes after reducing the effect of the radio interference by improving the transmitting and receiving power between nodes.

The effect of frequent nodes in the routing path on packet loss, when we examined the worst case where the publishing rate is eight msg/s. The PDR for is almost perfect when we have one publisher (avg. 10 readings: 0.99925) and two publishers (avg. 10 readings: 0.999625). At this point, there are no common nodes for the routing paths to Node 6, the broker node. In the case of 8 msg/s, the PDR starts drops clearly to (avg. 10 readings: 0.789045) when we have four publishers. While the PDR starts to drop clearly to (avg. 10 readings: 0.748308) when we have 16 publishers at the case of 4 msg/s, in the case of 2 msg/s, the worst PDR value is (avg. 10 readings: 0.894683) when we have the full traffic. For the cases when we have one msg/s, 0.5 msg/s and .25/msg/s, the lowest PDR is (avg. 10 readings: 0.9501) when we have full network traffic. Whereas, The PDR value is almost perfect for the rest cases even when we have full traffic network at 0.5 and .25 msg/s.

Table 17 :Network nodes values to enhance PDR

gain 0	1	-10.44	gain 0	9	-42.42
gain 1	0	-11.13	gain 9	0	-43.09
gain 0	2	-29.60	gain 0	10	-25.37
gain 2	0	-28.93	gain 10	0	-25.02
gain 0	3	-32.35	gain 0	11	-25.40
gain 3	0	-30.37	gain 11	0	-25.01
gain 0	4	-42.87	gain 0	12	-28.10
gain 4	0	-42.41	gain 12	0	-26.79
gain 0	5	-16.18	gain 0	13	-37.19
gain 5	0	-16.94	gain 13	0	-38.18
gain 0	6	-14.56	gain 0	14	-40.91
gain 6	0	-14.74	gain 14	0	-39.56
gain 0	7	-32.80	gain 0	15	-32.26
gain 7	0	-30.27	gain 15	0	-32.65
gain 0	8	-36.10	gain 0	16	-32.24
gain 8	0	-34.30	gain 16	0	-30.99

The common nodes in the routing path start to happen when we have four publishers and onward where the common nodes are node 12 in addition to the broker node 6. Tables 18

shows the number of times each node in the network is included in the routing paths, for all cases when we change the number of Publishers. Also, Figure 34 shows the results for performance metrics when we improved the radio signals to reduce the message drops.

For the case of four Publisher, Table 18(c) shows node 12 is included two times, and node 6 is included 4 times, for the routing paths of the four Publishers. At node 12, the node is a Publisher and at the same time is a relay node at the path for Publisher 24, through node 18. Therefore, node 12 is Publishing at the rate of 8 msg/s, and as a relay, it receives and forwards the messages from node 18 whenever they arrive and ready (processing time) to be sent to node 6 throughout the simulation time. At node 6, it is included four times in the routing paths, twice from node 12, one time from node 7 and 11; then it forwards the messages to the base station.

For detail messages count at the routing paths in the network during the simulation time, the path Publisher-ID:24 24-18-12-6-0: Node 24 sent 3996 messages in 500 seconds. Node 12 received 3911 messages from node 24. Node 12 received 3809 of the 3911 from node 18 which belongs to node 24. Node 6 received 3768 that belongs to node 24.

For the path, Publisher-ID:12 12-6-0: Node 12 Publishes 3997 messages in 500 seconds. Node 6 received 3819 that belongs to node 12.

For the path, Publisher-ID:20 20-16-11-6-0: Node 20 sent 3999 messages. Node 16 received all of them 3999. Then, Node 11 received from 16, 3990 messages. Only 10 messages are dropped.

For the path, Publisher-ID:4 4-8-7-6-0: Node 4 Publishes 4000 messages. Node 8 received 3998, only two dropped. Node 7 received 3981, dropped only 19. Node 6 received 3899, dropped 101 messages.

Table 18: The count of a common node in routing paths for all cases.

(a) One Publisher

Node(24):1	Node(19):0	Node(14):0	Node(9):0	Node(4):0
Node(23):0	Node(18):1	Node(13):0	Node(8):0	Node(3):0
Node(22):0	Node(17):0	Node(12):1	Node(7):0	Node(2):0
Node(21):0	Node(16):0	Node(11):0	Node(6):1	Node(1):0
Node(20):0	Node(15):0	Node(10):0	Node(5):0	

(b) Two Publishers

Node(24):1	Node(19):0	Node(14):0	Node(9):0	Node(4):0
Node(23):0	Node(18):1	Node(13):0	Node(8):0	Node(3):0
Node(22):0	Node(17):0	Node(12):1	Node(7):0	Node(2):0
Node(21):0	Node(16):1	Node(11):1	Node(6):2	Node(1):0
Node(20):1	Node(15):0	Node(10):0	Node(5):0	

(c) Four Publishers

Node(24):1	Node(19):0	Node(14):0	Node(9):0	Node(4):1
Node(23):0	Node(18):1	Node(13):0	Node(8):1	Node(3):0
Node(22):0	Node(17):0	Node(12):2	Node(7):1	Node(2):0
Node(21):0	Node(16):1	Node(11):1	Node(6):4	Node(1):0
Node(20):1	Node(15):0	Node(10):0	Node(5):0	

(d) Eight Publishers

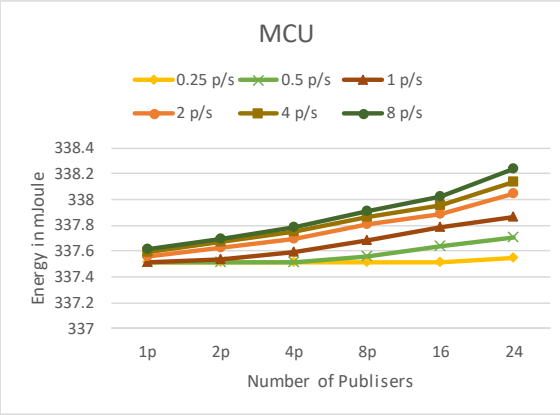
Node(24):1	Node(19):0	Node(14):1	Node(9):0	Node(4):1
Node(23):0	Node(18):1	Node(13):0	Node(8):2	Node(3):0
Node(22):1	Node(17):0	Node(12):2	Node(7):2	Node(2):1
Node(21):0	Node(16):2	Node(11):2	Node(6):8	Node(1):0
Node(20):1	Node(15):0	Node(10):1	Node(5):0	

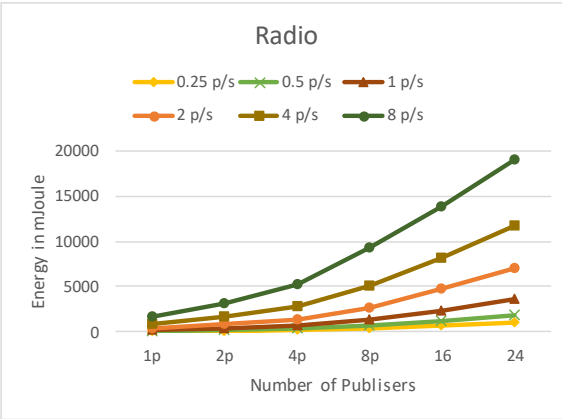
(e) Sixteen Publishers

Node(24):1	Node(19):0	Node(14):1	Node(9):0	Node(4):1
Node(23):0	Node(18):2	Node(13):1	Node(8):3	Node(3):0
Node(22):1	Node(17):1	Node(12):3	Node(7):5	Node(2):1
Node(21):0	Node(16):3	Node(11):5	Node(6):17	Node(1):0
Node(20):1	Node(15):0	Node(10):1	Node(5):1	

(f) Full publishers

Node(24):1	Node(19):1	Node(14):1	Node(9):1	Node(4):1
Node(23):1	Node(18):2	Node(13):2	Node(8):4	Node(3):1
Node(22):1	Node(17):2	Node(12):3	Node(7):8	Node(2):1
Node(21):1	Node(16):4	Node(11):8	Node(6):25	Node(1):1
Node(20):1	Node(15):1	Node(10):1	Node(5):2	





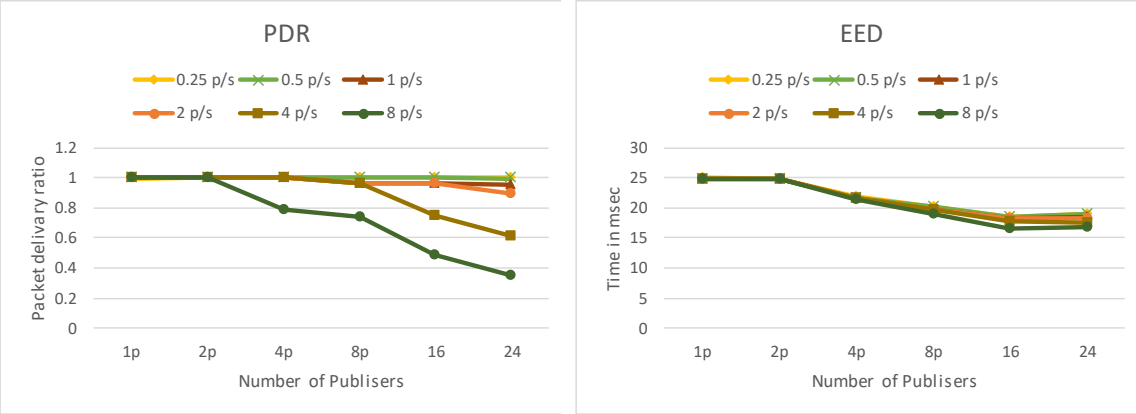


Figure 34: Results of TDDS with different publishing rates.

6.2 Availability support

In the definition of the Availability requirement for information security, which is the ability to ensure access and use of the information anytime and the system remains functioning. Denial-of-service attack (DoS) is one of the significant problems that disrupt the availability of the system; it occurs when an attacker takes action that prevents legitimate users from accessing targeted computer systems, devices or other network resources. DoS attacks flood the systems or networks with traffic to overwhelm the victim resources and make it difficult or impossible for legitimate users to use them.

6.2.1 Summary of DoS attacks and defenses:

DoS attacks and defenses can be categorized according to the layer protocols. In the physical layer, Jamming is the primary attack against WSNs. The Defense against it is through detecting and sleep or route around jammed regions. Although this will not prevent

a DoS attack, it could significantly increase the life of sensor nodes by reducing power consumption [87]. Link-layer threats include collisions, interrogation, and packet replay. An attacker might choose to execute a denial-of-sleep attack over a simple jamming-based DoS attack on a WSN to limit the attack's duration which prevents the radio from going into the sleep mode [88]. The defenses against link-layer threats include, authentication and anti-replay protection, detect and sleep, and broadcast attack protection.

In the network-layer, routing-disruption attacks can lead to DoS attacks in multihop sensor networks. The authors in [89] discuss sensor network routing vulnerabilities and attack countermeasures. General attacks on routing protocols include spoofing, replaying, or altering routing traffic. The link-layer authentication and antireplay can efficiently prevent these attacks.

Threats in the application-layer attack involve injecting false or replayed packets into the network at leaf nodes in a path-based DoS attack [90]. As the packet is forwarded to its destination, nodes along the path to the base station waste bandwidth and an energy transmitting the traffic. This attack can starve the network of legitimate traffic, because it consumes resources on the path to the base station, thus preventing other nodes from sending data to the base station. Combining packet authentication and anti-replay protection prevents these attacks.

6.2.2 Proposed enhancements and mitigations to the middleware

The network topology supported in our results squared-grid. Each node upon startup creates a list of its neighbor according to its given ID number. The maximum number of neighbor is eight for any node in the network. Therefore, each node is aware of its

neighbors and restricted it to receive messages from only the neighbors. In TinyOS communication, we can restrict to receive messages with a specific size. Any other messages with different size will be discarded. The lowest networking layer in TinyOS is called active messages (AM), which is implemented over the sensor's radio. It provides a single hop packet transmission and reception. An ID identifies each message, called the AM type, an 8-bit integer to identify the message type. Any other value of this ID of the received messages will be discarded. The size of exchanged messages is also known between the nodes, and so, upon receiving, we check the size, if it matches, the message will be processed, otherwise will be discarded. At the point where all above is checked, an attacker may pretend to be a neighbor node. However, the try will fail in the message authentication mechanism supported in our work along with the message integrity.

The payload in the middleware is represented as data-structure type that consists of several fields. The data fields include the origin node, which is the first node that generated a message. Also, the source and the destination data fields, that helps the middleware for routing and processing the messages received along the relay nodes. At first, when the message is generated, the origin field and the source field are the same value of the node ID. The source field will be changed in the relay nodes when the message travels toward the destination. Accordingly, when a relay node receives a message, it can check the source and the destination, and it will decide whether the message belongs to it or to decide the next hop should this message be forwarded to. The routing protocol used in our simulation is ID-Based routing, which only needs to know the destination node to decide what is the next hop according to the node ID (address). Also, an important data field in the payload which is the save the time-related values for the message. Tracking the message during the

whole network simulation execution is important. There are also other data fields, that are middleware specific such as, the topic type, message type, subject field, and sensor data readings field. The total size of the data fields is 16 bytes, the size of the message payload.

The basic middleware is subjected to routing disruption and replay attacks, as the message can be captured, and data fields can be modified. In our modified versions of the middleware, we can prevent these types of vulnerabilities. The whole payload message is encrypted and accounted for data confidentiality and integrity. Therefore, the information about the message routing fields cannot be modified without detection. The data fields in payload regarding the time can help us to prevent or mitigate the replay attacks. We can add a threshold time, larger than the maximum time for a message to travel from a direct neighbor hop.

Similarly, the value of the time-related field cannot be modified without detection. Therefore, we can check the time the messages received at the nodes, when it is larger than the threshold value, it will be assumed as a replay or redundant message, then discarded. There is a chance for a few redundant messages to go through the threshold period, but they can be filtered out later in the application level. The threshold period can be further reduced through more investigations.

In addition, we enhanced the middleware routing paths of the nodes that cause congestion in the network to mitigate the effect of traffic. Specifically, when we have full traffic, and all nodes are publishing messages, with a high publishing rate, many messages are dropped. The basic implementation of the middleware relies on having only one broker (server) node to work as a notification service. This node becomes a bottleneck in the system. All

published messages have to go through it to reach the Subscriber nodes. Figure 35 shows the distribution of the publisher nodes on the brokers available. The new enhancement is making the broker as a cluster of brokers. To investigate the effect of the new enhancement, we assigned Nodes 1,5, and 6 as the broker system. Before the enhancement, we tracked the messages from all publisher nodes and found the routing paths of all of them to reach the broker node. All common nodes are collected and shown in section 6.1 Table 18. Particular nodes become common for several routing paths. The chance of dropping packet for those nodes is very high when the publishing rate is high as in our experiments showed for publishing rate of 8 msg/s. As shown in Table 18(f), node 6 is common in 25 routing paths, nodes 11 and 7 are common to 8 paths. We made the nodes 1 and 5 as additional brokers along with node 6 in the original implementation. We can see the number of common routing paths is distributed among them. The routing paths are changed as we divided the nodes into three groups each will redirect their original paths to the closest broker. Figure 35 shows the distribution of the nodes. Table 19 shows the new routing paths for all nodes in the network after adding three brokers. This step results in significant improvements on the overall PDR of the network traffic. Some individual nodes, reached improvement of more than 30%. Table 20 shows the count of the common node for both brokers solutions.

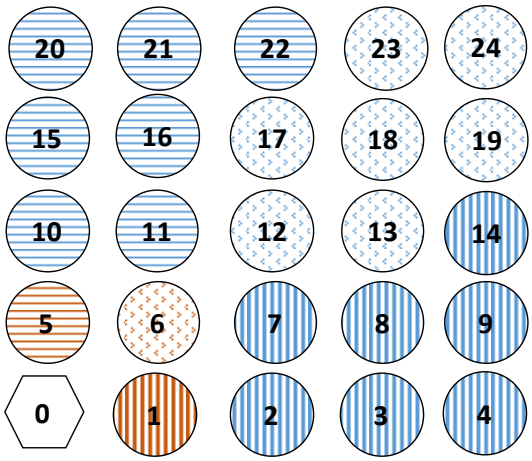


Figure 35: Three broker nodes, 1,5 and 6

The results of three brokers solutions are compared with the one broker solution, for the performance metrics PDR and EED. The worst case is selected when the publishing rate is 8 msg/s, where we have the most dropped packets compared to the lower rates. Figure 36 shows the comparison between the two solutions. The overall PDR for the three-broker solution is better the one broker solution. It results in increasing the overall EED while increasing the number of publishers. It confirms the previous results, when we increase the publishing rate, the number of dropped packets increases from the packet that is coming from the farther nodes, due to more hops they travel. Therefore, the closer nodes, contribute more in decreasing the overall average EED.

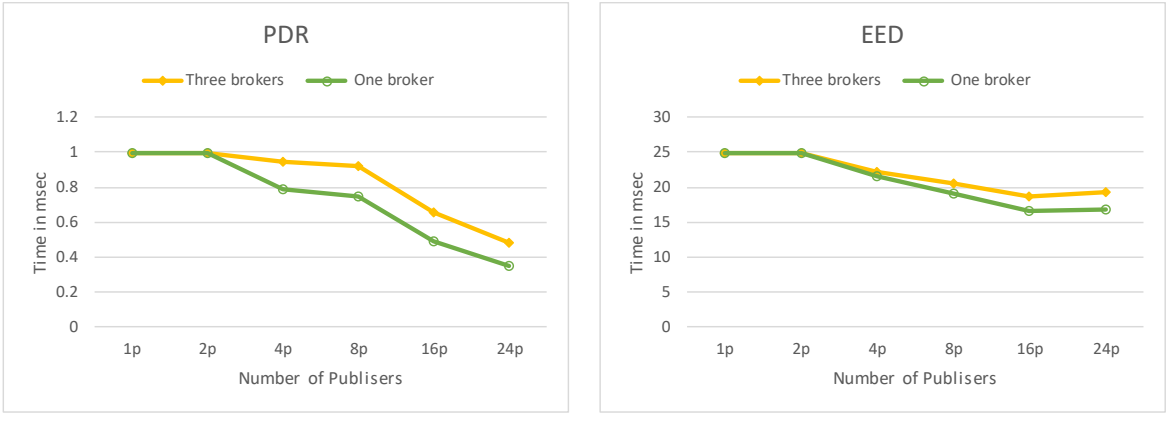


Figure 36: Comparison between broker solutions (8 msgs/s).

Table 19:Routing details for both brokers solutions

One broker (6)	Three brokers (1,5,6)
Publisher-ID:1 1-6-0 Hops=2	Publisher-ID:1 1-5-0 Hops=2
Publisher-ID:2 2-6-0 Hops=2	Publisher-ID:2 2-1-0 Hops=2
Publisher-ID:3 3-7-6-0 Hops=3	Publisher-ID:3 3-2-1-0 Hops=3
Publisher-ID:4 4-8-7-6-0 Hops=4	Publisher-ID:4 4-3-2-1-0 Hops=4
Publisher-ID:5 5-6-0 Hops=2	Publisher-ID:5 5-6-0 Hops=2
Publisher-ID:6 6-5-6-0 Hops=3	Publisher-ID:6 6-1-0 Hops=2
Publisher-ID:7 7-6-0 Hops=2	Publisher-ID:7 7-1-0 Hops=2
Publisher-ID:8 8-7-6-0 Hops=3	Publisher-ID:8 8-2-1-0 Hops=3
Publisher-ID:9 9-8-7-6-0 Hops=4	Publisher-ID:9 9-3-2-1-0 Hops=4
Publisher-ID:10 10-6-0 Hops=2	Publisher-ID:10 10-5-0 Hops=2
Publisher-ID:11 11-6-0 Hops=2	Publisher-ID:11 11-6-1-0 Hops=3
Publisher-ID:12 12-6-0 Hops=2	Publisher-ID:12 12-6-0 Hops=2
Publisher-ID:13 13-7-6-0 Hops=3	Publisher-ID:13 13-7-6-0 Hops=3
Publisher-ID:14 14-8-7-6-0 Hops=4	Publisher-ID:14 14-8-7-6-0 Hops=4
Publisher-ID:15 15-11-6-0 Hops=3	Publisher-ID:15 15-10-5-0 Hops=3
Publisher-ID:16 16-11-6-0 Hops=3	Publisher-ID:16 16-10-5-0 Hops=3
Publisher-ID:17 17-11-6-0 Hops=3	Publisher-ID:17 17-11-6-0 Hops=3
Publisher-ID:18 18-12-6-0 Hops=3	Publisher-ID:18 18-12-6-0 Hops=3
Publisher-ID:19 19-13-7-6-0 Hops=4	Publisher-ID:19 19-13-7-6-0 Hops=4
Publisher-ID:20 20-16-11-6-0 Hops=4	Publisher-ID:20 20-15-10-5-0 Hops=4
Publisher-ID:21 21-16-11-6-0 Hops=4	Publisher-ID:21 21-15-10-5-0 Hops=4
Publisher-ID:22 22-16-11-6-0 Hops=4	Publisher-ID:22 22-16-10-5-0 Hops=4
Publisher-ID:23 23-17-11-6-0 Hops=4	Publisher-ID:23 23-17-11-6-0 Hops=4
Publisher-ID:24 24-18-12-6-0 Hops=4	Publisher-ID:24 24-18-12-6-0 Hops=4

Table 20: The count of common node for both brokers solutions.

One broker (6)					Three brokers (1,5,6)				
(a) One Publisher									
Node(24):1	Node(19):0	Node(14):0	Node(9):0	Node(4):0	Node(24):1	Node(19):0	Node(14):0	Node(9):0	Node(4):0
Node(23):0	Node(18):1	Node(13):0	Node(8):0	Node(3):0	Node(23):0	Node(18):1	Node(13):0	Node(8):0	Node(3):0
Node(22):0	Node(17):0	Node(12):1	Node(7):0	Node(2):0	Node(22):0	Node(17):0	Node(12):1	Node(7):0	Node(2):0
Node(21):0	Node(16):0	Node(11):0	Node(6):1	Node(1):0	Node(21):0	Node(16):0	Node(11):0	Node(6):1	Node(1):0
Node(20):0	Node(15):0	Node(10):0	Node(5):0		Node(20):0	Node(15):0	Node(10):0	Node(5):0	
(b) Two Publishers									
Node(24):1	Node(19):0	Node(14):0	Node(9):0	Node(4):0	Node(24):1	Node(19):0	Node(14):0	Node(9):0	Node(4):0
Node(23):0	Node(18):1	Node(13):0	Node(8):0	Node(3):0	Node(23):0	Node(18):1	Node(13):0	Node(8):0	Node(3):0
Node(22):0	Node(17):0	Node(12):1	Node(7):0	Node(2):0	Node(22):0	Node(17):0	Node(12):1	Node(7):0	Node(2):0
Node(21):0	Node(16):1	Node(11):1	Node(6):2	Node(1):0	Node(21):0	Node(16):0	Node(11):0	Node(6):1	Node(1):0
Node(20):1	Node(15):0	Node(10):0	Node(5):0		Node(20):1	Node(15):1	Node(10):1	Node(5):1	
(c) Four Publishers									
Node(24):1	Node(19):0	Node(14):0	Node(9):0	Node(4):1	Node(24):1	Node(19):0	Node(14):0	Node(9):0	Node(4):1
Node(23):0	Node(18):1	Node(13):0	Node(8):1	Node(3):0	Node(23):0	Node(18):1	Node(13):0	Node(8):0	Node(3):1
Node(22):0	Node(17):0	Node(12):2	Node(7):1	Node(2):0	Node(22):0	Node(17):0	Node(12):2	Node(7):0	Node(2):1
Node(21):0	Node(16):1	Node(11):1	Node(6):4	Node(1):0	Node(21):0	Node(16):0	Node(11):0	Node(6):2	Node(1):1
Node(20):1	Node(15):0	Node(10):0	Node(5):0		Node(20):1	Node(15):1	Node(10):1	Node(5):1	
(d) Eight Publishers									
Node(24):1	Node(19):0	Node(14):1	Node(9):0	Node(4):1	Node(24):1	Node(19):0	Node(14):1	Node(9):0	Node(4):1
Node(23):0	Node(18):1	Node(13):0	Node(8):2	Node(3):0	Node(23):0	Node(18):1	Node(13):0	Node(8):1	Node(3):1
Node(22):1	Node(17):0	Node(12):2	Node(7):2	Node(2):1	Node(22):1	Node(17):0	Node(12):2	Node(7):1	Node(2):2
Node(21):0	Node(16):2	Node(11):2	Node(6):8	Node(1):0	Node(21):0	Node(16):1	Node(11):0	Node(6):3	Node(1):2
Node(20):1	Node(15):0	Node(10):1	Node(5):0		Node(20):1	Node(15):1	Node(10):3	Node(5):3	
(e) Sixteen Publishers									
Node(24):1	Node(19):0	Node(14):1	Node(9):0	Node(4):1	Node(24):1	Node(19):0	Node(14):1	Node(9):0	Node(4):1
Node(23):0	Node(18):2	Node(13):1	Node(8):3	Node(3):0	Node(23):0	Node(18):2	Node(13):1	Node(8):2	Node(3):1
Node(22):1	Node(17):1	Node(12):3	Node(7):5	Node(2):1	Node(22):1	Node(17):1	Node(12):3	Node(7):3	Node(2):3
Node(21):0	Node(16):3	Node(11):5	Node(6):17	Node(1):0	Node(21):0	Node(16):2	Node(11):2	Node(6):8	Node(1):6
Node(20):1	Node(15):0	Node(10):1	Node(5):1		Node(20):1	Node(15):1	Node(10):4	Node(5):4	
(f) Full publishers									
Node(24):1	Node(19):1	Node(14):1	Node(9):1	Node(4):1	Node(24):1	Node(19):1	Node(14):1	Node(9):1	Node(4):1
Node(23):1	Node(18):2	Node(13):2	Node(8):4	Node(3):1	Node(23):1	Node(18):2	Node(13):2	Node(8):2	Node(3):3
Node(22):1	Node(17):2	Node(12):3	Node(7):8	Node(2):1	Node(22):1	Node(17):2	Node(12):3	Node(7):4	Node(2):5
Node(21):1	Node(16):4	Node(11):8	Node(6):25	Node(1):1	Node(21):1	Node(16):2	Node(11):3	Node(6):11	Node(1):9
Node(20):1	Node(15):1	Node(10):1	Node(5):2		Node(20):1	Node(15):3	Node(10):6	Node(5):8	

6.3 Effective capacity and Energy consumption model

Environmental factors mainly cause the packet drop in wireless sensor networks. The source of the dropped packet happens at two layers, the physical layer, and MAC layer. At the physical-layer and in the absence of interfering transmissions, packet delivery performance is largely a function of the environment, the particular physical layer coding scheme, and individual receiver characteristics. Many factors have an impact on the packet

delivery. The environmental characteristics can cause multi-path signal reception or signal attenuation. The spatial separation between the sender and receiver can determine the received signal strength. Finally, minor variations in receiver and sender circuitry or battery levels can adversely impact on the physical layer functionality. At the medium access layer, interfering transmissions contribute to poor packet delivery performance. Many MAC layers contain mechanisms, such as carrier sense and link layer retransmissions, to counteract these effects. The application workload determines the traffic generated by nodes and hence the efficiency of channel access. Also, the topology affects how many nodes might potentially contend for the channel at a given point in time [91].

6.3.1 Sensors' effective capacity and publishing rates

The authors in [92, 93] show how to calculate the effective data capacities for sensor devices that are compliant with the IEEE 802.15.4 standard. Their work inspires the following discussion. We applied the equation that is related to our work.

The effective data capacity is defined as the maximum achievable data rate in the absence of any cross traffic. In the Carrier-sense multiple access with collision avoidance (CSMA-CA) scheme, effective data capacity is smaller than the data rate at the physical layer. The difference is due to channel access coordination to handle multiple, pipelined packets on the path, which incorporates the works of carrier sensing as well as random back-off mechanisms [93]. The MicaZ motes is a 2.4 GHz IEEE 802.15.4 compliant. It uses the Amtel ATMEGA128L micro-controller and ChipCon CC2420 IEEE 802.15.4 compliant radios chip. The duration of 2.4 GHz physical layers of 1 byte is 32 μs. Also, the link capacity at the physical layer is 250kbps. The maximum size of the Radio packet for CC2420 is 128 bytes including its headers and cyclic redundancy check (CRC).

When two nodes communicate in the CSMA-CA, the source node has to perform a clear channel assessment (CCA) to verify whether the medium is free or not. If the channel is free, the node will send out the data frame and wait for an acknowledge frame (optional). All other nodes, overhearing this communication, will defer their transmission. In the case of an occupied channel, an exponential backoff mechanism is used.

We need to calculate the effective data capacity for a single-hop connection between 2 neighbors. Therefore, to calculate the upper bound of the single-hop effective data capacity $C_{effective}$ we need to calculate the time needed between two frames, called T_{total}, as shown in equation (1). Which is the sum of the time needed for the headers overhead $T_{overhead}$, the waiting time T_{wait}, and the time needed to transmit the data payload $T_{data\_payload}$. There is an overhead regarding the optional acknowledge frames, which is of size 11 bytes.

The maximum size of the data payload is 128 bytes minus the size of the overhead, which 12 bytes for the head, and 2 bytes for CRC. Therefore, the size of the data payload is 114 bytes. The needed to transmit 114 bytes $T_{data\_payload} = 114 * (32\mu s) = 3.648ms$. Similarly, $T_{overhead} = 14 * (32\mu s) = 0.448ms$. The minimum wait time (e.g. from CCA time, radi,o turnaround time, and inter-frame interval) is $1.152ms$ by default of IEEE 802.15.4 standard. Therefore, $T_{total} = 3.648 + 0.448 + 1.152 = 5.248ms$. The $C_{effective}$ is calculated according to equation (2). $C_{physical}$ is equal to 250 kbps. Therefore, the upper bound of the single-hop effective data capacity is 173.78 kbps. This for the maximum allowed data payload 114 bytes. Similarly, the maximum effective capacity for other data payloads, 16 bytes, 32 bytes and 48 bytes is equal to 60.6 kbps, 97.56 kbps and 122.44 kbps, respectively.

$$T_{total} = T_{data_{payload}} + T_{overhead} + T_{wait} \quad (1)$$

$$C_{effective} \leq \frac{T_{data\_payload}}{T_{data\_payload} + T_{overhead} + T_{wait}} * C_{physical} \quad (2)$$

Moreover, due to the collision avoidance mechanism, the effective capacity of a wireless link decreases when there is more than one node within its collision domain. For example, when N active nodes, belonging to the same path, are within each other's transmission range, the maximum effective rate on that path is $\frac{C}{(N-1)}$ since only one of the N nodes can transmit at any time. In our simulation, the network topology is a square grid, and the number of nodes is 25.

In our experiments, we applied different publishing rate of the messages with a different number of publishers. The publishing rates are: 0.25, 0.5,1 ,2, 4, and 8 messages per second. The size of the payload of the messages is 16 bytes in addition to the 14 bytes of the packet header and CRC. Table 21 shows the capacity of each publishing rate. For example, to calculate the data rate of payload 16 bytes publishing rate is 4 msg/s, and the overhead bytes are 14 is equal to (16+14) * 4 (messages) * 8 (number of bits) = 960 bps.

To show why there are drop-in packets even when we applied the simulation parameter to get a perfect delivery ratio for only one publisher. Taking the worst case, when the network is full traffic, all nodes are publishing date with the same data rate.

Table 21: The corresponding date rate of the published messages.

Publishing rate (msg/s)	Data rate payload 16 Bytes (bps)
0.25	60
0.5	120
1	240
2	480
4	960
8	1920

An example to show the effect of the messages drops when we have a publishing rate of 8 msgs/s. Refer to discussion in the previous section for the one broker implementation. For the case of four Publisher, section 6.1 Table 18 shows node 12 is included 2 times, and node 6 is included 4 times, for the routing paths of the four Publishers. At node 12, the node is a Publisher and at the same time is A relay node at the path for Publisher 24, through the node 18. Therefore, node 12 is Publishing at the rate of 8 msg/s, and as a relay it receives and forwards the messages from node 18 whenever they arrive and ready (processing time) to be sent to node 6 throughout the simulation time. At node 6, it is included four times in the routing paths, twice from node 12, one time from node 7 and 11; then it forwards the messages to the base station. Therefore, we have 3 Nodes are forwarding packets to Node 6. Due to the collision avoidance mechanism, the effective capacity of the wireless link decreases when there is more than one node within its collision domain. Therefore, if we take the worst-case node, it would be node 6, but not all the node in the network are in the collision domain. Therefore, we considered the direct neighbors and the next direct neighbors. The total number of node are 15.

The upper bound of effective capacity for the 16 bytes packet load is equal to $\frac{60.6\ \text{kbps}}{(15-1)}$ = 4328.57 bps for each node. From Table 21, the capacity of the traffic for the 16 bytes packet payload is 1920 for each publisher. Therefore, the sum of capacity arriving to node 6 from 4 publishers is 4* 1920 = 7680 bps. As a result, we can see that the incoming packets speed is faster the capacity of the node to receive all the packets. The simulation results show the total PDR of the network is around 0.79. Most of the dropped packets are at the nodes 6. The PDR is not the same for all publishers. Our results showed in this example that the PDR of node 24 is 0.93, the PDR of node 20 is 0.87, the PDR of the node is 0.75, and finally, the PDR of node 4 is 0.59.

Similarly, when applying the calculation for the same case 4 publishers on the slower publishing rate, such as 4 msgs/s. From Table 21, the capacity of the traffic for the 16 bytes packet payload is 960 bps for each publisher. Therefore, the sum of capacity arriving at node 6 from 4 publishers is 4* 960 = 3840 bps, which is less than the maximum effective capacity. Our results show that the PDR, in this case, is 0.99975.

6.3.2 A mathematical model of radio energy consumption approximation

On this part, an approximation of the radio energy consumption to compare it with the simulation results in different cases of the number of Publishers in the network. In our experiments, the energy consumption is calculated by taking the summation of the energy consumption of all nodes in the network in milli-Joule (mJ) unit. The energy consumption equation is calculated using equation (3). The energy consumption depends on the state (Transmit, Receive, Idle, sleep) of the radio for the nodes. The sleep mode is when the

node is idle for a long time without any activity. Table 22, shows the values of the mode (state) of the radio $Current_{radio\_state}$ consumption, which are obtained from the datasheet of the MicaZ motes. $Duration_{radio\_state}$ represents the state duration the sensor spends, and V represents the constant voltage, which equal to 3 Volts.

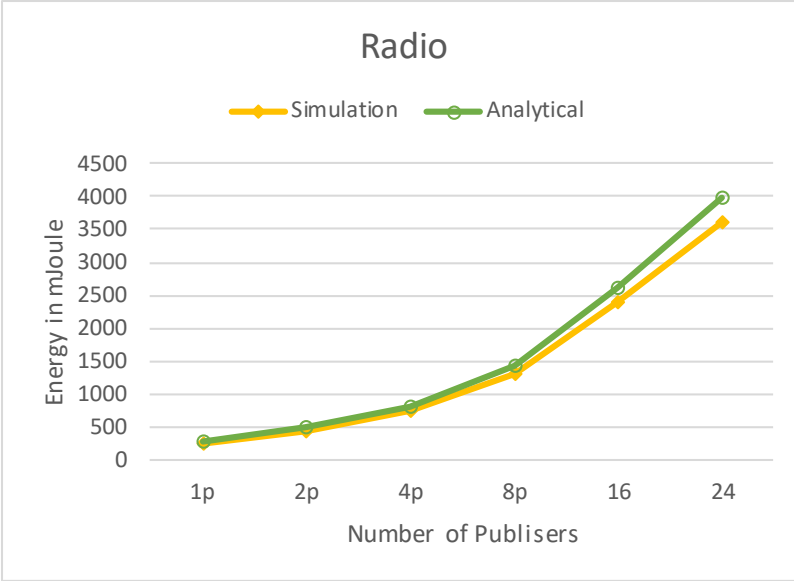
Table 22: Radio current consumption of MicaZ

MicaZ	
Mode	**Current**
Receive	19.7 mA
Transmit	17.4 mA
Idle	20 µA
Sleep	1 µA

$$Energy_{consumption} = \sum_{node}^{n} Duration_{radio\_state} * V_{sensor} * Current_{radio\_state} \quad (3)$$

In our simulation, we used the size of the message in addition to the overhead as explained, is equal to 30 bytes. Also, the duration of transmitting 1 Byte is 32 µs. Therefore, the duration of time to send one message in our experiments is equal to $30B * 32\mu s = 0.96ms$. We will assume in the calculations that PDR is perfect. Therefore, our calculations represent the upper bound of the energy consumption. For different publishing rates, the number of messages varies accordingly. In the 1 Publisher case, where the publishing rate is 1 msg/s, the number of messages sent publisher node ID_24 is 500 messages. The messages will follow the routing path as explained on section 6.1 Table 16. Also, section 6.2 Table 20 provides the number of nodes that are involved according to the number of the publishers' case, and how many times the messages are transmitted and received during the routing path. Thus, for the case of 1 Publisher, node 24 transmits 500, and the nodes in between (18-12-6) receive and transmit (forward) the messages to node 0, so the number

of states in transmit mode are 4 as well as the receive states, which the sum of all values in Table 20. We can apply the results on equation (3). The variable $Duration_{radio\_state}$, is the number of messages (500) multiply the sum of states (4) taken from Table 20, and then multiply time duration of 1 message ($0.96ms$). Directly we can apply the transmitting state and receiving state in equation (3). When the sending is done, the node goes to the idle mode. Therefore, the total $Duration_{idle}$ is calculated as the total simulation time minus the total duration time in the transmitting and receiving state. For the sleeping state, the nodes that are not involved at any activity (publishing, subscribing or relay) is considered in the sleep state. Which is the number of zeros in the Table 20. Thus, the $Duration_{idle}$ is the entire simulation time (500 seconds). The number of nodes at sleeping mode for example in the case of 1 publishers is 20 nodes, (case of 2 publishers is 17 nodes), and the number of nodes at the case of the full publisher is 0. The total energy is calculated by taking the summation of each state. Figure 37 shows the comparison between the simulation and analytical results of the Radio energy consumption.



	1p	2p	4p	8p	16	24
Simulation	236.1062	434.1966	731.2997	1308.329	2380.518	3621.591
Analytical	273.6346	482.6616	798.5059	1432.876	2602.64	3978.972

Figure 37: Radio consumption, simulation vs analytical results

6.4 Space (Memory) complexity

Space complexity is a measure of the amount of the total space is used by the algorithm to complete the execution concerning the input size. It is calculated for how much memory is needed in the worst case at any point in the algorithm. The concern is with how the space needs grow, in big-Oh terms, as the size N of the input problem grows. The space complexity is divided into constant space and auxiliary space. The constant space is the one which is fixed for the algorithm, generally equals to the space used by the input and the local variables. Whereas, the auxiliary space, is the temporary space used by the algorithm during the execution. Usually, the constant memory is ignored and only focus on the auxiliary space to compare the performance of the algorithms.

Regardless of the architecture of the machine, the space complexity of a variable is considered to be equal to one memory unit. Therefore, the space complexity required for an array variable is equal to the size of the array.

6.4.1 Multilinear Modular Hashing (MMH) algorithm

General properties of the MMH algorithm is that the memory unit considered is of size 32bits. Also, the prime integer used is $p = 2^{32} + 15$, which is the least of formula suggested in the theory by Carter and Wegman, to implement a division-less modular reduction. It can be in the range of $[2^{32} ; 2^{32} + 2^{16}]$. It is used to implement the division-less modular reduction. The output of the function is a 32-bit integer.

A vector of different 32-bit numbers used to implement the so-called inner-product operation. A good choice is to take only prime numbers. The number of elements of such a vector must be at least equal to the key length divided by 4. The reduction operation is to perform the modulus of the prime number. The pseudocode of the MMH algorithm is as follows:

```
MMH(msg, key, n)
    SumHigh = SumLow = 0
    For i = 1 to n
        load msg[i]
        load key[i]
        (ProdHigh, ProdLow = msg[i] * key[i]
         SumLow = SumLow + ProdLow
         SumHigh = SumHigh + ProdHigh + carry
    Reduce (SumHigh, SumLow) mod 2^32 + 15 and then mod 2^32
```

The space complexity of the algorithm is as follows:

The variable msg and key are arrays of size n. All the individual variables are of size 32 bits (4 Bytes). The constant complexity: n Bytes+ n Bytes+ 4 Bytes (variable n) + 8 Bytes (2 variables, SumHigh and Sumlow). Therefore, the total in bytes is (2 n + 12) Bytes. In big-Oh terms, it is equal to $O(n)$. The auxiliary complexity: we have 3 temporary variables, that are reused in the loop. Therefore, the total is 12 Bytes. In big-Oh terms is equal to $O(1)$. The total space complexity = Constant space + Auxiliary space= (2n+12) +12= 2n+24.

Table 23 shows the required data needed when we change the input size to the algorithm implementation. We examined two sensor platforms, MicaZ and TelosB. Figure 38 shows linear trending of the results taken from Table 23, which is similar to the memory complexity analysis above. The RAM results are matching the analysis.

Table 23: MMH Algorithm implementation results on sensor memory.

	MicaZ			Telosb	
Bytes	**RAM**	**ROM**		**RAM**	**ROM**
16	**36**	**2074**		**38**	**302**
32	**68**	**2106**		**66**	**334**
48	**100**	**2138**		**102**	**366**
64	**132**	**2170**		**134**	**398**
80	**164**	**2202**		**166**	**430**
96	**196**	**2234**		**198**	**462**
112	**228**	**2266**		**230**	**494**
128	**260**	**2298**		**262**	**526**
144	**292**	**2330**		**294**	**558**

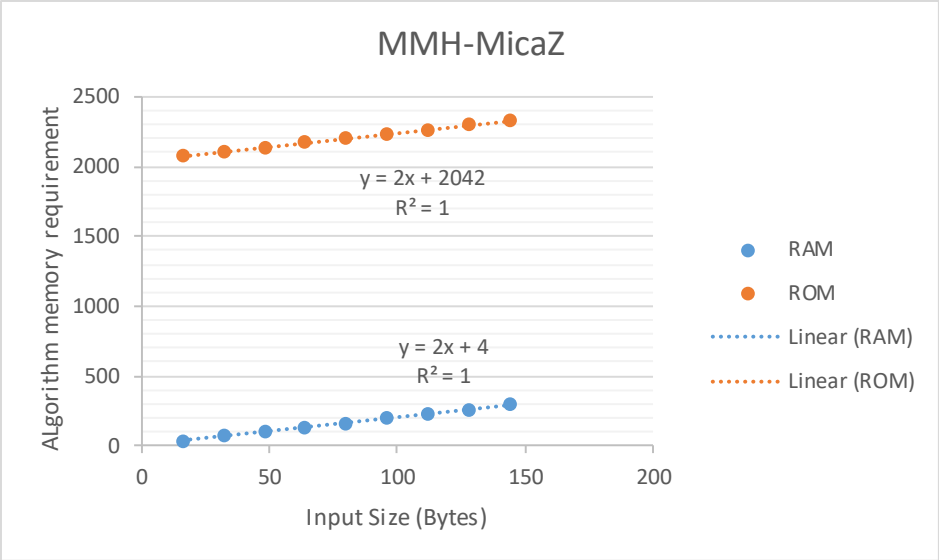


Figure 38: MMH Input memory vs the required memory complexity

6.4.2 Secure Hash Algorithm-1 (SHA-1)

The algorithm takes an input of any size and produces a 160-bit (20-byte) hash value. The pseudocode of the SHA-1 algorithm is as follows:

```
SHA-1(msg, n)
h0 = 0x67452301; h1 = 0xEFCDAB89; h2 = 0x98BADCFE; h3 = 0x10325476; h4 = 0xC3D2E1F0
ml = message length in bits
for each chunk (512 bits size)
   break chunk into sixteen 32-bit big-endian words w[i], 0 ≤ i ≤ 15
   Extend the sixteen 32-bit words into eighty 32-bit words:
   for i from 16 to 79
      w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1

   Initialize hash value for this chunk:
   a = h0;    b = h1;     c = h2;     d = h3;    e = h4;

   Main loop:
   for i from 0 to 79
      if 0 ≤ i ≤ 19 then
         f = (b and c) or ((not b) and d)
         k = 0x5A827999
      else if 20 ≤ i ≤ 39
         f = b xor c xor d
         k = 0x6ED9EBA1
      else if 40 ≤ i ≤ 59
         f = (b and c) or (b and d) or (c and d)
```

```
        k = 0x8F1BBCDC
    else if 60 ≤ i ≤ 79
        f = b xor c xor d
        k = 0xCA62C1D6

    temp = (a leftrotate 5) + f + e + k + w[i]
    e = d;   d = c;       c = b leftrotate 30;       b = a;       a = temp;

  h0 = h0 + a;   h1 = h1 + b;     h2 = h2 + c;     h3 = h3 + d;    h4 = h4 + e;

hh = (h0 leftshift 128) or (h1 leftshift 96) or (h2 leftshift 64) or (h3 leftshift 32) or h4 // The hash result in hh
```

The space complexity analysis for the SHA-1 algorithm is as follows: The variable msg is an array of size n. All variables of size 32 bits (4 Bytes), except ml of size 64 bits (8 Bytes units) and hh of size 160 bits (20 Bytes). The constant complexity: n Bytes (for the msg array) + 4 Bytes (for variable n) + 28 Bytes (for the initial variables, 5 of size 4B memory units (variables $h0 - h4$) and 1 of size 8B (variable ml)+ 20 Bytes (for variable hh)). Therefore, the total in bytes is (n+52) Bytes. In big-Oh terms, it is equal to $O(n)$. The auxiliary complexity: we have some temporary variables: Array variable w of size 320 Bytes, 80 entries of size 4 Bytes each. Also, 9 variables and their total size is 36 Bytes ($(a - e), i, f, k, temp$). Therefore, the total is 356 Bytes (320+36). In big-Oh terms is equal to $O(1)$. The total space complexity = Constant space + Auxiliary space= (n +52) +356= n +408.

For both algorithms, there is a direct relation with RAM. The difference from the ROM is the constant value in the equation. The contribution is from the source code, and the size of constants values and the defined variables. Increasing the size array will increase the memory size. Figure 39 shows linear trending of the results taken from Table 24, which is

similar to the memory complexity analysis above. The RAM results are matching the analysis.

Table 24: SHA1 Algorithm implementation results on sensor memory.

	MicaZ			Telosb	
Bytes	**RAM**	**ROM**		**RAM**	**ROM**
16	**397**	**2126**		**398**	**1896**
32	**413**	**2142**		**414**	**1912**
48	**429**	**2158**		**430**	**1928**
64	**445**	**2174**		**446**	**1944**
80	**461**	**2190**		**462**	**1960**
96	**477**	**2206**		**478**	**1976**
112	**493**	**2222**		**494**	**1992**
128	**509**	**2238**		**510**	**2008**
144	**525**	**2254**		**526**	**2024**

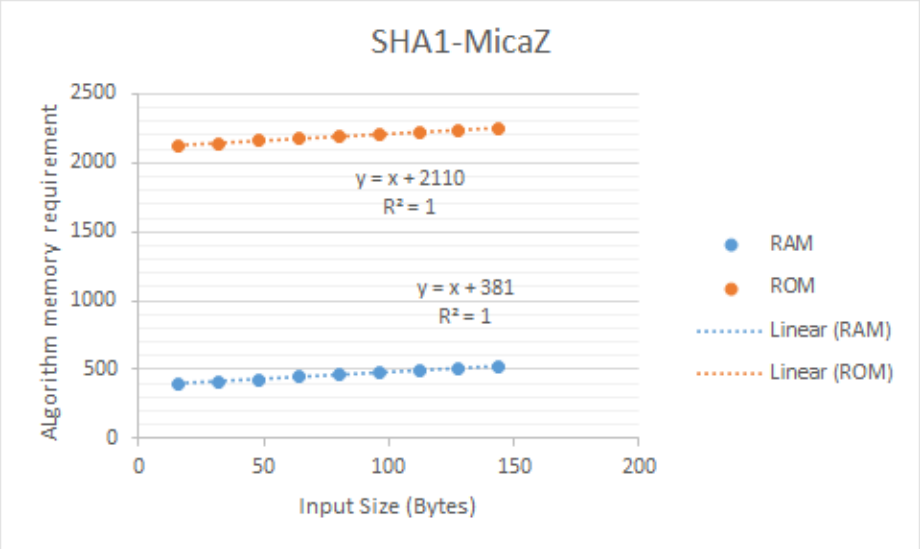


Figure 39: SHA1 Input memory vs the required memory complexity

6.5 Packet loss analysis

This section introduces the problem of the individual readings of the nodes, especially the nodes with symmetric paths to the base station. All reported readings for the performance metrics have been collected for the entire nodes in the network, without taking the

individual readings of the nodes. However, we tracked the reading of the individual reading of the nodes as well. We noticed that there are some nodes with symmetric path to the base station have differences in the packet delivery ratio readings with a noticeable reading difference as nearly 30%. The symmetric path means that the targeted nodes have a similar number of hops and distances to the base station. Therefore, it is expected that they have similar reading and not necessarily identical values. Section 6.1 Table 14 shows parameters to specify the topology of our network to generate the values for the nodes in the first column, and a description of the parameters, in the second columns. It is an example of the values from which the simulator generates the gain values for the network. Several files for the nodes gain are generated, to ensure randomness with the readings. Although some nodes are in the symmetric path to the base station, their corresponding gain values are not identical.

Table 25 shows the average values of 15 files gain readings and provide the cross-table readings of the nodes in the network. It shows the gain between the nodes. The colored squares in the table to show the value of the symmetric nodes 4 and node 20 matched hops. The matched color shows corresponding value in the path of the nodes toward the base-station. For example, the first hope node of the nodes 4 and 20 is 8 and 16, respectively. The gain value for the node pairs 4:8 and 20:16 is (-17.15) dBm and (-16.5487) dBm, respectively. Similarly, we can track the rest of the path to the base-station. We followed the same routing detail for the individual nodes discussed in section 6.1 Table 16.

Table 25 : Cross-table gain reading of the average of 15 files.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	0	-11.1253	-23.568	-35.148	-38.694	-9.2326?	-17.8107	-26.3787	-34.602	-39.5773	-24.1587	-27.354	-33.002	-37.6273	-41.74	-31.9687	-34.3473	-36.6727	-40.3373	-43.7047	-38.112	-40.1413	-41.4887	-43.8547	-45.448
1	-11.1253	0	-10.9087	-24.8467	-31.024	-19.2407	-10.01	-17.6253	-27.3487	-34.87	-26.3407	-24.7753	-26.562	-30.5787	-35.8433	-34.0133	-32.344	-35.3567	-36.748	-39.3	-38.6987	-39.6147	-39.3813	-40.0727	-44.6907
2	-23.568	-10.9087	0	-9.838	-24.356	-27.584	-17.922	-9.94467	-16.5807	-28.006	-30.7707	-25.9327	-25.9327	-31.4707	-36.2453	-35.0693	-32.872	-33.6547	-37.026	-40.534	-39.9727	-38.688	-38.046	-41.0673	
3	-35.148	-24.8467	-9.838	0	-10.7493	-35.8927	-25.7013	-17.7773	-10.1227	-18.1387	-36.8727	-31.818	-27.22	-24.6347	-26.9533	-40.4527	-37.6593	-34.9207	-34.392	-32.652	-44.7833	-40.344	-38.8753	-39.5807	-39.034
4	-38.694	-31.024	-24.356	-10.7493	0	-40.1173	-32.9333	-25.6333	-17.15	-10.4613	-40.85	-35.97	-31.644	-26.578	-23.6847	-43.0227	-40.5593	-36.9767	-34.2873	-35.1227	-45.8333	-42.4153	-40.4607	-38.8447	-38.8713
5	-9.23267	-19.2407	-27.584	-35.8927	-40.1173	0	-10.2307	-25.5607	-31.4653	-38.1873	-10.152	-17.6167	-27.0307	-33.8713	-40.23	-23.7247	-25.7207	-32.3427	-36.2373	-40.808	-32.3507	-33.332	-37.1447	-38.6913	-41.8613
6	-17.8107	-10.01	-17.922	-25.7013	-32.9333	-10.2307	0	-10.7027	-24.7847	-33.7827	-16.3053	-10.528	-16.4507	-26.7327	-33.4573	-27.168	-24.0687	-26.8533	-32.3087	-34.478	-33.542	-31.7927	-34.444	-37.452	-40.2893
7	-26.3787	-17.6253	-9.94467	-17.7773	-25.6333	-25.5607	-10.7027	0	-10.6073	-24.2293	-27.5313	-17.726	-10.6673	-18.2553	-26.7373	-30.774	-27.626	-23.9967	-26.914	-30.456	-35.2233	-34.0253	-32.6373	-32.7887	-36.0967
8	-34.602	-27.3487	-16.5807	-10.1227	-17.15	-31.4653	-24.7847	-10.6073	0	-12.0333	-33.4547	-27.912	-17.138	-10.1893	-17.2587	-36.0193	-30.332	-25.784	-24.706	-27.252	-39.766	-35.456	-33.2213	-32.926	-33.9207
9	-39.5773	-34.87	-28.006	-18.1387	-10.4613	-38.1873	-33.7827	-24.2293	-12.0333	0	-39.566	-34.5587	-27.936	-17.4713	-10.8593	-40.9907	-38.264	-30.9327	-26.8167	-24.226	-42.5507	-41.2693	-37.3387	-33.574	-32.942
10	-24.1587	-26.3407	-30.7707	-36.8727	-40.85	-10.152	-16.3053	-27.5313	-33.4547	-39.566	0	-10.386	-25.0227	-33.2253	-37.6493	-11.3353	-19.5107	-26.592	-34.2547	-38.348	-23.6087	-27.3247	-32.406	-36.214	-41.8887
11	-27.354	-24.7753	-25.9327	-31.818	-35.97	-17.6167	-10.528	-17.726	-27.912	-34.5587	-10.386	0	-10.3373	-25.378	-33.14	-17.6473	-10.1807	-18.3533	-27.546	-31.4993	-27.34	-24.5233	-27.982	-31.652	-36.1613
12	-33.002	-26.562	-23.9487	-27.22	-31.644	-27.0307	-16.4507	-10.6673	-17.138	-27.936	-25.0227	-10.3373	0	-10.7173	-23.2867	-26.124	-15.97	-8.716	-16.658	-27.024	-32.7367	-27.3287	-25.918	-24.99	-32.7667
13	-37.6273	-30.5787	-25.0107	-24.6347	-26.578	-33.8713	-26.7327	-18.2553	-10.1893	-17.4713	-33.2253	-25.378	-10.7173	0	-11.4693	-33.5347	-26.6807	-18.2293	-11.7387	-17.5567	-34.9733	-32.082	-25.834	-22.168	-27.6093
14	-41.74	-35.8433	-31.4707	-26.9533	-23.6847	-40.23	-33.4573	-26.7373	-17.2587	-10.8593	-37.6493	-33.14	-23.2867	-11.4693	0	-39.306	-33.2053	-28.516	-17.7967	-10.7967	-40.0067	-37.4033	-33.168	-25.778	-23.5393
15	-31.9687	-34.0133	-36.2453	-40.4527	-43.0227	-23.7247	-27.168	-30.774	-36.0193	-40.9907	-11.3353	-17.6473	-26.124	-33.5347	-39.306	0	-9.474	-24.8287	-33.94	-38.3533	-9.62667	-17.1333	-26.108	-34.016	-38.8267
16	-34.3473	-32.344	-35.0693	-37.6593	-40.5593	-25.7207	-24.0687	-27.626	-30.332	-38.264	-19.5107	-10.1807	-15.97	-26.6807	-33.2053	-9.474	0	-10.0127	-23.9787	-32.2247	-16.5487	-9.39133	-17.654	-27.938	-33.4907
17	-36.6727	-35.3567	-32.872	-34.9207	-36.9767	-32.3427	-26.8533	-23.9967	-25.784	-30.9327	-26.592	-18.3533	-8.716	-18.2293	-28.516	-24.8287	-10.0127	0	-9.30067	-24.1273	-27.4067	-16.3887	-11.4587	-17.0167	-25.2193
18	-40.3373	-36.748	-33.6547	-34.392	-34.2873	-36.2373	-32.3087	-26.914	-24.706	-26.8167	-34.2547	-27.546	-16.658	-11.7387	-17.7967	-33.94	-23.9787	-9.30067	0	-9.73867	-33.358	-26.8353	-17.8773	-9.73667	-16.7493
19	-43.7047	-39.3	-37.026	-32.652	-35.1227	-40.808	-34.478	-30.456	-27.252	-24.226	-38.348	-31.4993	-27.024	-17.5567	-10.7967	-38.3533	-32.2247	-24.1273	-9.73867	0	-39.6707	-33.018	-27.0073	-17.1893	-10.4913
20	-38.112	-38.6987	-40.534	-44.7833	-45.8333	-32.3507	-33.542	-35.2233	-39.766	-42.5507	-23.6087	-27.34	-32.7367	-34.9733	-40.0067	-9.62667	-16.5487	-27.4067	-33.358	-39.6707	0	-9.65067	-25.3107	-31.77	-37.7333
21	-40.1413	-39.6147	-39.9727	-40.344	-42.4153	-33.332	-31.7927	-34.0253	-35.456	-41.2693	-27.3247	-24.5233	-27.3287	-32.082	-37.4033	-17.1333	-9.39133	-16.3887	-26.8353	-33.018	-9.65067	0	-10.846	-24.288	-31.9593
22	-41.4887	-39.3813	-38.688	-38.8753	-40.4607	-37.1447	-34.444	-32.6373	-33.2213	-37.3387	-32.406	-27.982	-25.918	-32.7667	-33.168	-26.108	-17.654	-11.4587	-17.8773	-27.0073	-25.3107	-10.846	0	-10.8633	-24.7187
23	-43.8547	-40.0727	-38.046	-39.5807	-38.8447	-38.6913	-37.452	-32.7887	-32.926	-33.574	-36.214	-31.652	-24.99	-22.168	-25.778	-34.016	-27.938	-17.0167	-9.73867	-17.1893	-31.77	-24.288	-10.8633	0	-8.50333
24	-45.448	-44.6907	-41.0673	-39.034	-38.8713	-41.8613	-40.2893	-36.0967	-33.9207	-32.942	-41.8887	-36.1613	-32.7667	-27.6093	-23.5393	-38.8267	-33.4907	-25.2193	-16.7493	-10.4913	-37.7333	-31.9593	-24.7187	-8.50333	0

6.5.1 Results and Analysis

In this subsection, we focus on the case where we got inconsistent results regarding the symmetric nodes, and to investigate if other factors can control the misbehavior of the values in the initial results. The focus in the following study is on the example of symmetric nodes 4 and 20 as stated in the introduction of section 6.5. The path of node 4 is 8-7-6-0 with a number of hops equal to 4. The path for node 20 is16-11-6-0 with number of hops equal to 4. We increased the message rate for the publisher node to make the messages drop obvious when we have only two nodes are publishing in the network. We have a message rate of 16 messages per second (msg/s) and 32 msg/s. We reduced the simulation time to 100 seconds instead of 500 seconds to get the results faster. The total number of messages that are sent by each publisher is equal to 1600 messages and 3200 messages for message rates of 16 msg/s and 32 msg/s, respectively. Table 26 shows the detail results of the experiment where the message rate is 16 msg/s for two publishers, node 4 and node 20. Similarly, Table 27 Table 26 shows the detail results of the experiment where the message rate is 32 msg/s. As stated in the introduction, node 6 is the broker node, where the messages are collected from the network nodes then passed to the base-station.

Table 26: Detail table for Publishing rate 16 msg/s

Readings	Node 4		Node 20		N4-N20	Node 4		Node 20		N4-N20	PDR N4	PDR N20	PDR N4	PDR N20
#	Sent	Received	Sent	Rec @ BS	Rec @ BS	Rec @ 6	Snt 6	Rec @ 6	Snt 6	Rec @ 6			PDR @ 6	PDR @ 6
1	1600	1063	1597	1308	-245	1511	1511	1432	1432	79	0.664375	0.819036	0.944375	0.896681
2	1600	1086	1597	1311	-225	1518	1518	1450	1450	68	0.67875	0.820914	0.94875	0.907952
3	1600	1090	1597	1285	-195	1496	1496	1443	1443	53	0.68125	0.804634	0.935	0.903569
4	1600	1128	1597	1295	-167	1524	1524	1433	1433	91	0.705	0.810895	0.9525	0.897307
5	1600	1081	1597	1335	-254	1525	1525	1459	1459	66	0.675625	0.835942	0.953125	0.913588
6	1600	1087	1597	1331	-244	1510	1510	1461	1461	49	0.679375	0.833438	0.94375	0.91484
7	1600	1092	1597	1299	-207	1506	1506	1451	1451	55	0.6825	0.8134	0.94125	0.908579
8	1600	1073	1597	1302	-229	1515	1515	1467	1467	48	0.670625	0.815279	0.946875	0.918597
9	1600	1080	1597	1332	-252	1503	1503	1459	1459	44	0.675	0.834064	0.939375	0.913588
10	1600	1101	1597	1311	-210	1494	1494	1462	1462	32	0.688125	0.820914	0.93375	0.915466
11	1600	1096	1597	1315	-219	1505	1505	1455	1455	50	0.685	0.823419	0.940625	0.911083
12	1600	1062	1597	1316	-254	1493	1493	1439	1439	54	0.66375	0.824045	0.933125	0.901064
13	1600	1105	1597	1313	-208	1506	1506	1431	1431	75	0.690625	0.822167	0.94125	0.896055
14	1600	1087	1597	1315	-228	1495	1495	1450	1450	45	0.679375	0.823419	0.934375	0.907952
15	1600	1085	1597	1318	-233	1491	1491	1452	1452	39	0.678125	0.825297	0.931875	0.909205
Average	1600	1087.733	1597	1312.4	-224.667	1506.133	1506.133	1449.6	1449.6	56.53333	0.679833	0.821791	0.941333	0.907702

Table 27: Detail table for Publishing rate 32 msg/s

Readings	Node 4		Node 20		N4-N20	Node 4		Node 20		N4-N20	PDR N4	PDR N20	PDR N4	PDR N20
#	Sent	Received	Sent	Rec @ BS	Rec @ BS	Rec @ 6	Snt 6	Rec @ 6	Snt 6	Rec @ 6			PDR @ 6	PDR @ 6
1	3200	2083	3194	2521	-438	2937	2937	2856	2856	81	0.650938	0.789292	0.917813	0.894177
2	3200	2144	3194	2508	-364	2946	2946	2868	2868	78	0.67	0.785222	0.920625	0.897934
3	3200	2132	3194	2549	-417	2948	2948	2868	2868	80	0.66625	0.798059	0.92125	0.897934
4	3200	2109	3194	2521	-412	2957	2957	2863	2863	94	0.659063	0.789292	0.924063	0.896368
5	3200	2147	3194	2557	-410	2977	2977	2907	2907	70	0.670938	0.800564	0.930313	0.910144
6	3200	2160	3194	2562	-402	2966	2966	2913	2913	53	0.675	0.802129	0.926875	0.912023
7	3200	2098	3194	2560	-462	2935	2935	2903	2903	32	0.655625	0.801503	0.917188	0.908892
8	3200	2134	3194	2563	-429	2944	2944	2882	2882	62	0.666875	0.802442	0.92	0.902317
9	3200	2109	3194	2560	-451	2935	2935	2889	2889	46	0.659063	0.801503	0.917188	0.904508
10	3200	2109	3194	2539	-430	2928	2928	2896	2896	32	0.659063	0.794928	0.915	0.9067
11	3200	2115	3194	2521	-406	2964	2964	2883	2883	81	0.660938	0.789292	0.92625	0.90263
12	3200	2131	3194	2530	-399	2963	2963	2881	2881	82	0.665938	0.79211	0.925938	0.902004
13	3200	2085	3194	2521	-436	2938	2938	2877	2877	61	0.651563	0.789292	0.918125	0.900751
14	3200	2084	3194	2587	-503	2949	2949	2890	2890	59	0.65125	0.809956	0.921563	0.904822
15	3200	2138	3194	2516	-378	2968	2968	2880	2880	88	0.668125	0.787727	0.9275	0.901691
Average	3200	2118.533	3194	2541	-422.467	2950.333	2950.333	2883.733	2883.733	66.6	0.662042	0.795554	0.921979	0.90286

The table shows 15 readings corresponding to the 15 gain files for the simulator as explained in section 6.1. Each row in the table provides the number of the messages sent and received for the publishers at the base-station, and the broker node. Also, it shows the corresponding packet delivery ratio (PDR) at the broker and the base-station. There are two columns (N4-N20) where we provide the difference in the message received at the base-station and broker between the publisher 4 and publisher 20. The number of messages received from publisher 20 is subtracted from the messages received from publisher 4 at broker and base-station. We notice that the values in the difference-column regarding the base-station are always in a negative sign. The negative sign in the column indicates that the messages received from node 20 are more than the messages received from node 4. This the reason we made the study to investigate this behavior, assuming that the simulator favors specific nodes although they are symmetric, and the randomness is provided through the gain files. The sign in the difference column may indicate the correctness of the results regarding fairness and randomness if the sign is changed alternately for a given number of readings (rows in the table).

The observations that we can extract from table 4 that the number of messages received from publisher 20 is more than the messages received from publisher 4 which results in the PDR of publisher 20 is more than PDR of publisher 4 at the base-station. The difference in the PDR is significant (nearly 14%) although both nodes published the same number of nodes in favor of node 20. Also, the difference in the PDR is less at the broker (nearly 3.4%) in favor of node 4. Similar observations are concluded when we increase the message rate to 32 msg/s, but the difference in the PDR is slightly less, nearly 13% at base-station and nearly 2% at the broker.

At this point, we have a node (Publisher 20) that suffers less from message drop than the other node (Publisher 4). It was assumed that the problem was due to the difference in the gain path through the base-station. However, after providing more gain files, the average value of the gain files shows only a slight difference in the overall value when we compare the two publishers' paths. In [94], the author added a mechanism to minimize the effect of co-channel interference. The mechanism is simply set different spacing time between the nodes when they started publishing and called it interference-free-scheduling (IFS) algorithm. The author made a study to compare the results after adding this algorithm, which showed PDR improved by nearly 3.5 times for the studied scenarios in their work.

In our example, the IFS algorithm is applied by making the Publisher 20 to start sending data after 200 ms from Publisher 4. To check for the symmetry comparison between the publishers, we need to put them on the same conditions. Thus, they should start all at the same time, and cancel the job of IFS algorithm. We verified the observation by swapping the conditions and made Publisher 4 to start 200 ms after Publisher 20. All the results are

interchanged as provided in Table 28 for the message rate of 16 ms/s. Similarly, the same observations are provided in Table 29 for the message rate of 32 msg/s.

Table 28: Detail table for Publishing rate 16 msg/s (swap conditions)

Readings	Node 4		Node 20		N4-N20	Node 4		Node 20		N4-N20	PDR N4	PDR N20	PDR N4	PDR N20
#	Sent	Received	Sent	Rec @ BS	Rec @ BS	Rec @ 6	Snt 6	Rec @ 6	Snt 6	Rec @ 6			PDR @ 6	PDR @ 6
1	1597	1302	1600	1094	208	1448	1448	1492	1492	-44	0.815279	0.68375	0.9067	0.9325
2	1597	1325	1600	1091	234	1460	1460	1507	1507	-47	0.829681	0.681875	0.914214	0.941875
3	1597	1314	1600	1063	251	1452	1452	1495	1495	-43	0.822793	0.664375	0.909205	0.934375
4	1597	1326	1600	1095	231	1460	1460	1516	1516	-56	0.830307	0.684375	0.914214	0.9475
5	1597	1318	1600	1075	243	1447	1447	1501	1501	-54	0.825297	0.671875	0.906074	0.938125
6	1597	1299	1600	1088	211	1442	1442	1511	1511	-69	0.8134	0.68	0.902943	0.944375
7	1597	1309	1600	1077	232	1451	1451	1483	1483	-32	0.819662	0.673125	0.908579	0.926875
8	1597	1295	1600	1063	232	1455	1455	1498	1498	-43	0.810895	0.664375	0.911083	0.93625
9	1597	1275	1600	1143	132	1424	1424	1512	1512	-88	0.798372	0.714375	0.891672	0.945
10	1597	1320	1600	1088	232	1448	1448	1522	1522	-74	0.82655	0.68	0.9067	0.95125
11	1597	1296	1600	1103	193	1435	1435	1517	1517	-82	0.811522	0.689375	0.89856	0.948125
12	1597	1289	1600	1118	171	1438	1438	1510	1510	-72	0.807138	0.69875	0.900438	0.94375
13	1597	1303	1600	1051	252	1442	1442	1483	1483	-41	0.815905	0.656875	0.902943	0.926875
14	1597	1297	1600	1097	200	1441	1441	1493	1493	-52	0.812148	0.685625	0.902317	0.933125
15	1597	1287	1600	1071	216	1449	1449	1497	1497	-48	0.805886	0.669375	0.907326	0.935625
Average	1597	1303.667	1600	1087.8	215.8667	1446.133	1446.133	1502.467	1502.467	-56.3333	0.816322	0.679875	0.905531	0.939042

Table 29: Detail table for Publishing rate 32 msg/s (swap conditions).

Readings	Node 4		Node 20		N4-N20	Node 4		Node 20		N4-N20	PDR N4	PDR N20	PDR N4	PDR N20
#	Sent	Received	Sent	Rec @ BS	Rec @ BS	Rec @ 6	Snt 6	Rec @ 6	Snt 6	Rec @ 6			PDR @ 6	PDR @ 6
1	3194	2557	3200	2078	479	2887	2887	2934	2934	-47	0.800564	0.649375	0.903882	0.916875
2	3194	2533	3200	2107	426	2859	2859	2919	2919	-60	0.793049	0.658438	0.895116	0.912188
3	3194	2528	3200	2107	421	2886	2886	2961	2961	-75	0.791484	0.658438	0.903569	0.925313
4	3194	2528	3200	2112	416	2873	2873	2925	2925	-52	0.791484	0.66	0.899499	0.914063
5	3194	2497	3200	2137	360	2876	2876	2951	2951	-75	0.781778	0.667813	0.900438	0.922188
6	3194	2547	3200	2160	387	2895	2895	2940	2940	-45	0.797433	0.675	0.906387	0.91875
7	3194	2541	3200	2100	441	2867	2867	2941	2941	-74	0.795554	0.65625	0.897621	0.919063
8	3194	2560	3200	2121	439	2896	2896	2931	2931	-35	0.801503	0.662813	0.9067	0.915938
9	3194	2535	3200	2147	388	2878	2878	2961	2961	-83	0.793676	0.670938	0.901064	0.925313
10	3194	2511	3200	2129	382	2860	2860	2959	2959	-99	0.786162	0.665313	0.895429	0.924688
11	3194	2539	3200	2140	399	2874	2874	2965	2965	-91	0.794928	0.66875	0.899812	0.926563
12	3194	2532	3200	2121	411	2895	2895	2933	2933	-38	0.792736	0.662813	0.906387	0.916563
13	3194	2536	3200	2096	440	2879	2879	2939	2939	-60	0.793989	0.655	0.901378	0.918438
14	3194	2544	3200	2090	454	2890	2890	2942	2942	-52	0.796493	0.653125	0.904822	0.919375
15	3194	2569	3200	2127	442	2890	2890	2943	2943	-53	0.804321	0.664688	0.904822	0.919688
Average	3194	2537.133	3200	2118.133	419	2880.333	2880.333	2942.933	2942.933	-62.6	0.794344	0.661917	0.901795	0.919667

The next step to study the symmetric publishers, we started all publishers at the same time, canceling the job of IFS algorithm. The results are provided in Table 30. It is clear from the table that the PDR values are almost the same. We can notice the sign of the difference column, which the sigh is alternating. Meaning that there is no favoring the messages received from specific publishers as it was assumed at the beginning. Furthermore, the total number of messages received at the base-station is less than the total number of the

messages received when the IFS algorithm is presented. Similarly, the same observations are provided in Table 31 for the message rate of 32 msg/s.

Table 30: Detail table for Publishing rate 16 msg/s (same conditions)

Readings	Node 4		Node 20		N4-N20	Node 4		Node 20		N4-N20	PDR N4	PDR N20	PDR N4	PDR N20
#	Sent	Received	Sent	Rec @ BS	Rec @ BS	Rec @ 6	Snt 6	Rec @ 6	Snt 6	Rec @ 6			PDR @ 6	PDR @ 6
1	1600	1021	1600	1060	-39	1430	1430	1464	1464	-34	0.638125	0.6625	0.89375	0.915
2	1600	1045	1600	1036	9	1456	1456	1454	1454	2	0.653125	0.6475	0.91	0.90875
3	1600	1039	1600	1038	1	1446	1446	1431	1431	15	0.649375	0.64875	0.90375	0.894375
4	1600	1043	1600	1071	-28	1442	1442	1440	1440	2	0.651875	0.669375	0.90125	0.9
5	1600	1038	1600	1047	-9	1447	1447	1446	1446	1	0.64875	0.654375	0.904375	0.90375
6	1600	1047	1600	1038	9	1432	1432	1425	1425	7	0.654375	0.64875	0.895	0.890625
7	1600	1043	1600	1039	4	1436	1436	1432	1432	4	0.651875	0.649375	0.8975	0.895
8	1600	1059	1600	1018	41	1436	1436	1442	1442	-6	0.661875	0.63625	0.8975	0.90125
9	1600	1025	1600	1025	0	1452	1452	1439	1439	13	0.640625	0.640625	0.9075	0.899375
10	1600	1057	1600	1043	14	1434	1434	1433	1433	1	0.660625	0.651875	0.89625	0.895625
11	1600	1061	1600	1028	33	1437	1437	1431	1431	6	0.663125	0.6425	0.898125	0.894375
12	1600	1038	1600	1063	-25	1410	1410	1438	1438	-28	0.64875	0.664375	0.88125	0.89875
13	1600	1043	1600	1043	0	1437	1437	1453	1453	-16	0.651875	0.651875	0.898125	0.908125
14	1600	1031	1600	1052	-21	1437	1437	1457	1457	-20	0.644375	0.6575	0.898125	0.910625
15	1600	1044	1600	1052	-8	1445	1445	1439	1439	6	0.6525	0.6575	0.903125	0.899375
Average	1600	1042.267	1600	1043.533	-1.26667	1438.467	1438.467	1441.6	1441.6	-3.13333	0.651417	0.652208	0.899042	0.901

Table 31: Detail table for Publishing rate 32 msg/s (same conditions).

Readings	Node 4		Node 20		N4-N20	Node 4		Node 20		N4-N20	PDR N4	PDR N20	PDR N4	PDR N20
#	Sent	Received	Sent	Rec @ BS	Rec @ BS	Rec @ 6	Snt 6	Rec @ 6	Snt 6	Rec @ 6			PDR @ 6	PDR @ 6
1	3200	2078	3200	2030	48	2875	2875	2816	2816	59	0.649375	0.634375	0.898438	0.88
2	3200	2052	3200	2079	-27	2879	2879	2878	2878	1	0.64125	0.649688	0.899688	0.899375
3	3200	2124	3200	2069	55	2885	2885	2870	2870	15	0.66375	0.646563	0.901563	0.896875
4	3200	2052	3200	2035	17	2868	2868	2870	2870	-2	0.64125	0.635938	0.89625	0.896875
5	3200	2062	3200	2088	-26	2871	2871	2859	2859	12	0.644375	0.6525	0.897188	0.893438
6	3200	2077	3200	2015	62	2878	2878	2866	2866	12	0.649063	0.629688	0.899375	0.895625
7	3200	2051	3200	2107	-56	2894	2894	2857	2857	37	0.640938	0.658438	0.904375	0.892813
8	3200	2088	3200	2025	63	2877	2877	2832	2832	45	0.6525	0.632813	0.899063	0.885
9	3200	2057	3200	2090	-33	2864	2864	2891	2891	-27	0.642813	0.653125	0.895	0.903438
10	3200	2051	3200	2017	34	2858	2858	2866	2866	-8	0.640938	0.630313	0.893125	0.895625
11	3200	2105	3200	2118	-13	2874	2874	2874	2874	0	0.657813	0.661875	0.898125	0.898125
12	3200	2082	3200	2043	39	2841	2841	2844	2844	-3	0.650625	0.638438	0.887813	0.88875
13	3200	2046	3200	2073	-27	2860	2860	2872	2872	-12	0.639375	0.647813	0.89375	0.8975
14	3200	2081	3200	2109	-28	2859	2859	2874	2874	-15	0.650313	0.659063	0.893438	0.898125
15	3200	2086	3200	2049	37	2893	2893	2847	2847	46	0.651875	0.640313	0.904063	0.889688
Average	3200	2072.8	3200	2063.133	9.666667	2871.733	2871.733	2861.067	2861.067	10.66667	0.64775	0.644729	0.897417	0.894083

To test for more publishers, we increased the number of publishers to be 4 and made the same experiment setups. We kept the publishers 4 and 20 as a symmetric pair in addition to publishers 12 and 24 as depicted in our network topology. Following the same analysis of the tables, we can see from Table 31 that the PDR of the symmetric nodes is close. The PDR of publisher 4 is 0.489, and the PDR of publisher 20 is 0.483. The PDR of the other publishers is different, for publisher 12 is 0.579, and publisher 24 is 0.459 which is less,

due to the routing path of node 24 has more hops than node 12. As stated before, having more hops results in suffering from more dropping of messages along the path. The same observations are presented for a message rate of 32 msg/s. We added a graph to show the frequency of the messages that are successfully arrived at the broker node, for 20 seconds intervals for the entire simulation. Figure 40 shows the frequency of the four publishers. We can see that for the symmetric pair we focus on, the publishers 4 and 20 have close values.

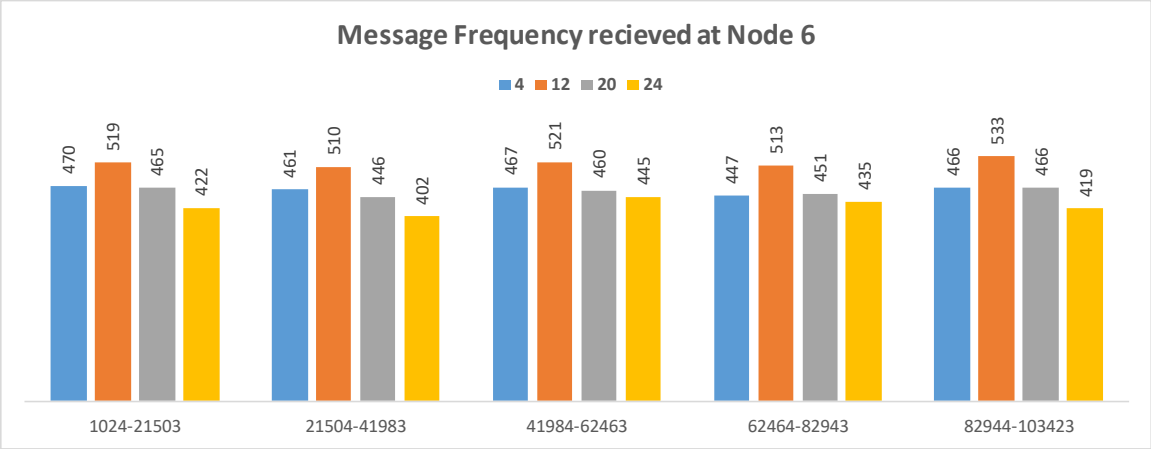


Figure 40: Message frequency received at node 6 for message rate 32 msg/s.

Extended results for nodes (4,20),(14,22)

In this part, we extended the PDR analysis to account for scenarios where the nodes start exactly at the same time instead of applying IFS algorithm. We found that the symmetric nodes are similar in the PDR value, unlike when we used the IFS algorithm. However, the overall PDR is better when we apply IFS. Table 41 shows the PDR values for the individual nodes at the base-station and the broker node. On each experiment, we also calculated the difference of the count of the successfully received messages between the symmetric pairs,

at the base-station node in order to show the randomness and the differences in each experiment, as the number of received messages is not fixed.

Figure 41: Four publishers (4,20,14,22) with message rate16 msg/s (all same time).

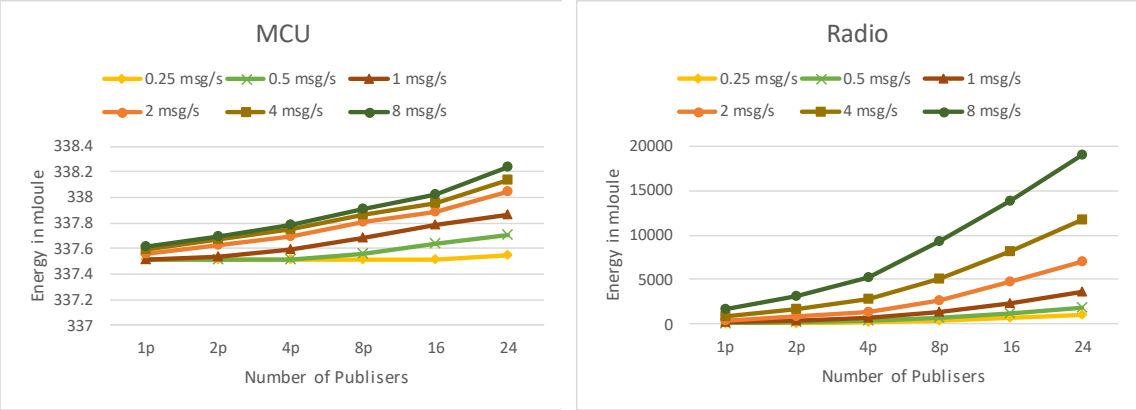
Readings	Node 4		14		Node 20		22		PDR N4	PDR N14	PDR N20	PDR N22	N4-N20	N14-N22
#	Sent	Rec @ BS	Sent	Rec @ BS	Sent	Rec @ BS	Sent	Rec @ BS					Rec @ BS	Rec @ BS
1	1600	531	1600	574	1600	523	1600	531	0.331875	0.35875	0.326875	0.331875	8	43
2	1600	555	1600	529	1600	544	1600	529	0.346875	0.330625	0.34	0.330625	11	0
3	1600	538	1600	534	1600	496	1600	572	0.33625	0.33375	0.31	0.3575	42	-38
4	1600	524	1600	532	1600	573	1600	543	0.3275	0.3325	0.358125	0.339375	-49	-11
5	1600	529	1600	541	1600	481	1600	544	0.330625	0.338125	0.300625	0.34	48	-3
6	1600	521	1600	530	1600	555	1600	535	0.325625	0.33125	0.346875	0.334375	-34	-5
7	1600	533	1600	564	1600	572	1600	510	0.333125	0.3525	0.3575	0.31875	-39	54
8	1600	536	1600	542	1600	539	1600	531	0.335	0.33875	0.336875	0.331875	-3	11
9	1600	517	1600	564	1600	511	1600	513	0.323125	0.3525	0.319375	0.320625	6	51
10	1600	523	1600	525	1600	574	1600	520	0.326875	0.328125	0.35875	0.325	-51	5
11	1600	529	1600	543	1600	478	1600	548	0.330625	0.339375	0.29875	0.3425	51	-5
12	1600	567	1600	544	1600	537	1600	525	0.354375	0.34	0.335625	0.328125	30	19
13	1600	559	1600	541	1600	533	1600	540	0.349375	0.338125	0.333125	0.3375	26	1
14	1600	521	1600	528	1600	560	1600	484	0.325625	0.33	0.35	0.3025	-39	44
15	1600	560	1600	534	1600	554	1600	520	0.35	0.33375	0.34625	0.325	6	14
Average	1600	536.2	1600	541.6667	1600	535.3333	1600	529.6667	0.335125	0.338542	0.334583	0.331042	-4.7	18.9

For the next part, we applied a Poisson distribution for the data rate and maintained the same overall average data rates. We investigated changing the time the symmetric nodes start publishing data. The values of the PDR of the symmetric nodes are almost the same, unlike when we apply the IFS algorithm. Also, the overall PDR is improved. Table 32, shows the results the nodes publish data at the same, PDR is close for all publishers and better than previous results without Poisson message rate. However, N14-N22 is always negative, due to a number of messages sent in favor of node 20.

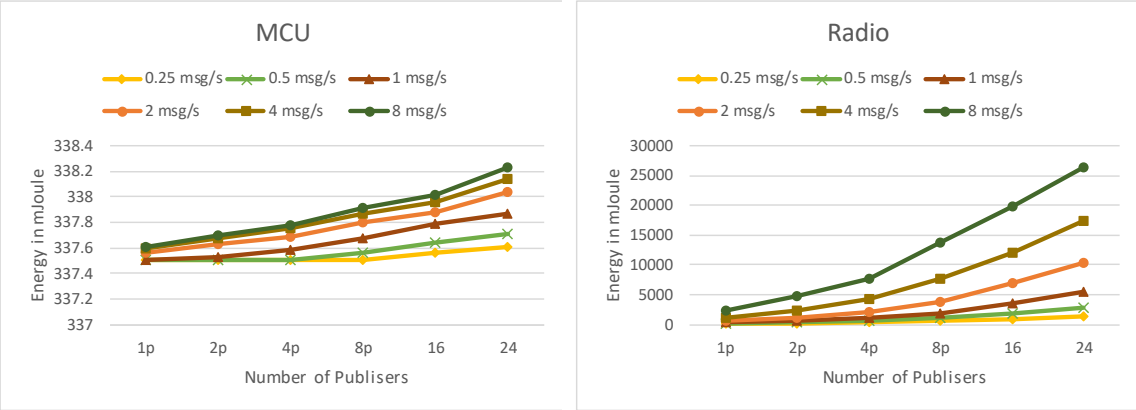
Table 32: Four publishers with Poisson message rate16 msg/s (same time)

Readings	Node 4		14		Node 20		22		PDR N4	PDR N14	PDR N20	PDR N22	N4-N20	N14-N22
#	Sent	Rec @ BS	Sent	Rec @ BS	Sent	Rec @ BS	Sent	Rec @ BS					Rec @ BS	Rec @ BS
1	1517	943	1430	892	1459	948	1632	1064	0.621622	0.623776	0.64976	0.651961	-5	-172
2	1517	993	1430	870	1459	940	1632	1055	0.654581	0.608392	0.644277	0.646446	53	-185
3	1517	950	1430	920	1459	909	1632	1033	0.626236	0.643357	0.623029	0.632966	41	-113
4	1517	922	1430	894	1459	947	1632	1063	0.607779	0.625175	0.649075	0.651348	-25	-169
5	1517	945	1430	897	1459	894	1632	1069	0.62294	0.627273	0.612748	0.655025	51	-172
6	1517	921	1430	916	1459	931	1632	1060	0.607119	0.640559	0.638108	0.64951	-10	-144
7	1517	998	1430	892	1459	945	1632	1026	0.657877	0.623776	0.647704	0.628676	53	-134
8	1517	965	1430	883	1459	930	1632	1073	0.636124	0.617483	0.637423	0.657475	35	-190
9	1517	960	1430	900	1459	926	1632	1031	0.632828	0.629371	0.634681	0.63174	34	-131
10	1517	947	1430	900	1459	919	1632	1042	0.624258	0.629371	0.629883	0.63848	28	-142
11	1517	965	1430	921	1459	941	1632	1062	0.636124	0.644056	0.644962	0.650735	24	-141
12	1517	965	1430	901	1459	915	1632	1062	0.636124	0.63007	0.627142	0.650735	50	-161
13	1517	973	1430	902	1459	933	1632	1022	0.641397	0.630769	0.639479	0.626225	40	-120
14	1517	946	1430	885	1459	929	1632	1021	0.623599	0.618881	0.636737	0.625613	17	-136
15	1517	981	1430	876	1459	927	1632	1037	0.646671	0.612587	0.635367	0.635417	54	-161
Average	1517	958.2667	1430	896.6	1459	928.9333	1632	1048	0.631685	0.626993	0.636692	0.642157	32.5	-146

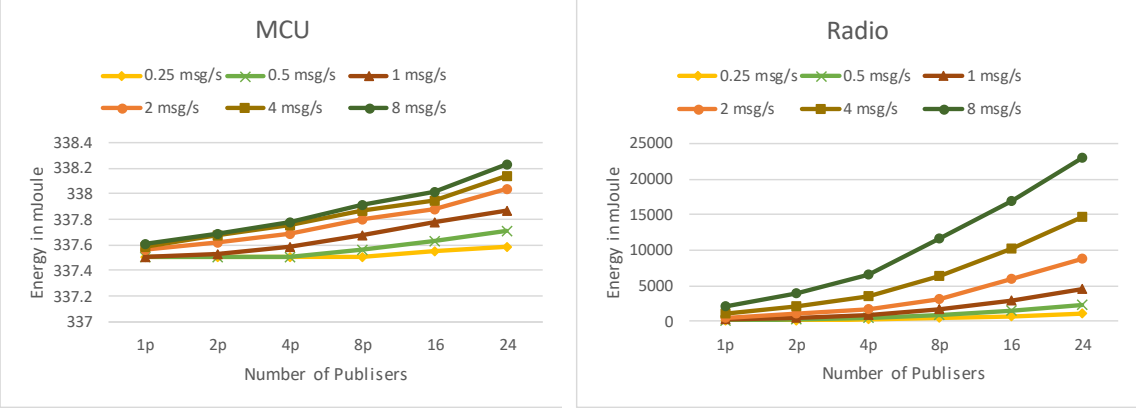
We repeated the experiments for different average data rate using Poisson message rate for the same performance metrics observed in Chapter 5. We conducted additional simulation runs where we change the average data rate from 0.25 to 8 msg/s to study the effect of the traffic density in the network for all versions of the middleware. As shown in Figure 42, the results regarding MCU are almost identical for all versions of the middleware as we stated earlier. Also, for the readings of each middleware version, there is a slight increment in the energy consumption for MCU while we are changing the data rate from 0.25 to 8 msg/s. For the Radio energy consumption part, the effect of applying different data rates is evident following the same trend for all middleware versions. In every data rate option, TDDS_AES_MMH consume less energy than TDDS_AES_SHA1.



(a) Results for TDDS: MCU and Radio



(b) Results for TDDS_AES_SHA1: MCU and Radio

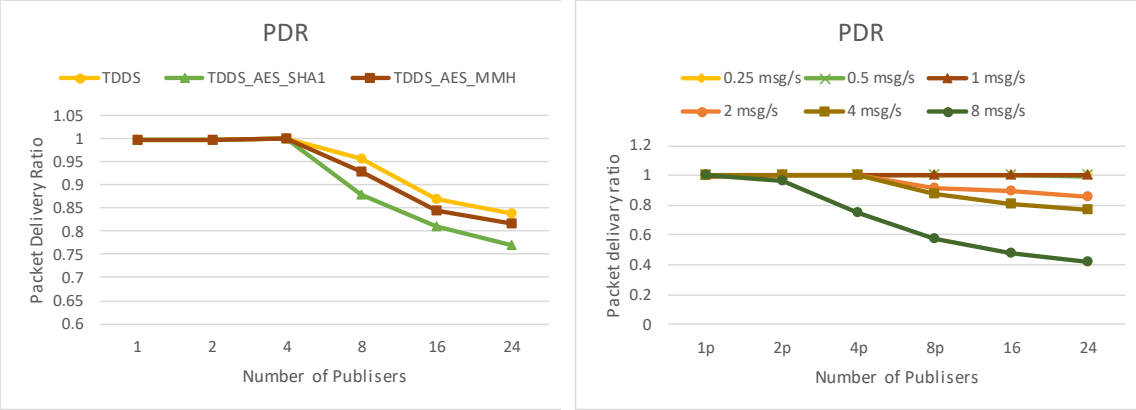


(c) Results for TDDS_AES_MMH: MCU and Radio

Figure 42: Energy consumption results for all MW versions with different data rates.

Figure 43 provides a comparison of PDR value for different data rates and different middleware versions. Figure 43 part(a) shows the comparison for three versions of the

middleware at an average data rate equal to 4 msg/s. We can notice that PDR for all version is almost perfect when the number of the publishers is less than or equal to 4. The configuration of the channels and radio parameters for the simulator are set up so that without interference from neighbors, the probability of dropped message is very small. However, when the number of publishers is eight or higher, the effect of congestion of the messages starts to take effect on the base-station and in the routing path relay nodes to the base-station. The message drop is at the highest point in the case of 24 publishers. Also, we can notice that TDDS_AES_MMH performs better than TDDS_AES_SHA1. Figure 43 part(b) compares the results obtained from changing the data rate in average from 0.25 msg/s to 8 msg/s for all cases for the same version of the middleware TDDS_AES_SHA1. It is clear from the figure that the PDR is almost perfect for slow data rates less than 2 msg/s on average. The PDR value decreases when the data rate increases and the packet loss is very high when the data rate reaches 8 msg/s.



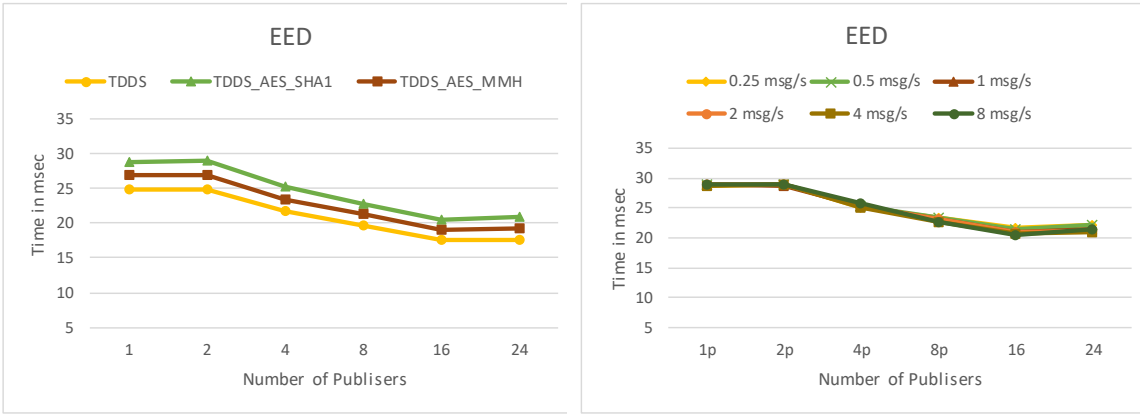
(a) Different MW versions at the data rate 4 msg/s **(b) TDDS_AES_SHA1 at different data rates**

Figure 43: Packet delivery ratio comparisons.

Figure 44 provides a comparison of EED value when the data rate is 4 msg/s. Figure 44 part(a) shows the comparison between three main implementations, the basic unsecured

middleware, and the other two secured implementations. We can see that the delay decreases as the number of Publishers increases from 1 to 16. The location of the Publisher affects the delay. The farther the distance of the Publisher from the Subscriber the longer the time packets need to arrive. In the first case, the Publisher is at the far end in the network from the Subscriber. In the second case, with the two Publishers, including the same publisher from the first case, we can notice that the value EED is close to the first case. The reason why, the number of hops for the messages to reach the subscriber node is the same in both cases, as discussed in the routing detail for the network. In the case of 24 Publishers, which is full load traffic, the delay roughly the same as the previous case. For the last case, eight new publishers are added to the previous case and distributed in symmetrical locations far from the subscriber node. We made individual calculations of each node, in the network, and found that the total average EED time of the new publishers does not contribute to reducing the overall EDD of the network. In the other cases, when we move from case to case, we double the number of the publishers each time, this way, the contribution on the overall EDD will be noticeable. Also, we can notice that the secured versions follow the same trend of basic TDDS version, but in all cases, it takes longer time due to extra computations for security components. Furthermore, Figure 44 part(b) shows the values of EDD for different data rates. The values are nearly the same in all cases, but for faster rates, there is more drop-in packets especially from the far nodes from the subscriber. We noticed after analyzing the individual nodes values that there are more messages successfully arrived closer to the subscriber than the farther nodes, which contributes on reducing the overall network EED value on faster data rates. This case can be noticed for the cases of 16 publishers and 24 publishers. In the ideal case, where we do

not have drop packets in each case or equally expected drops, the value of EED does not change with the change in data rate.



(a) Different MW versions at the data rate 4 msg/s **(b) TDDS_AES_SHA1 at different data rates**

Figure 44: End-to-End delay comparisons.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

WSN is an important platform that offers a variety of applications that can be used with significant interests in both academia and industry. However, developing applications for WSNs is a complex process, especially when interacting with the sensor's hardware components, and dealing with the constraints of WSNs regarding memory requirements, the computation power, and energy resources. Therefore, there is a need for an intermediate layer to facilitate the process of focusing on developing the sensor applications without worrying about the underlying hardware components, where the importance of supporting a middleware is emerged. In this work, a light-weight middleware called TinyDDS is considered for improvements in the aspect of security measures. It is investigated using simulation experiments to evaluate the performance when adding information security components. The middleware has ported the DDS features and functionalities into WSN, which includes many categories of QoS policies easy to use. The publish/subscribe communication model in DDS is well suited for large-scale distributed computing systems. That is because of the decoupling properties for the network's participants in time, space, and synchronization. DDS is a standard for supporting real-time distributed systems based on the publish/subscribe scheme.

The security enhancements in the middleware aim to provide data confidentiality, integrity, and message authentication. Several cryptographic algorithms are surveyed to select suitable algorithms to be implemented in our modified version of TinyDDS to account for the resource constraints in sensor nodes. Our security module is transparent to the middleware, that is there are no modifications to the original application or the communication protocol. The results regarding performance evaluation for energy consumption, memory occupancy and time delay, show a reasonable overhead in comparison with the basic middleware. We explained further results to reduce the overhead when implementing the integrity functionality.

7.2 Future work directions

As a future direction work, we can investigate other cryptographic algorithms and the trade-offs in their security and their impact on the WSN performance.

For the results, we have focused on having one subscriber in the network, which is also acting as the base-station. Furthermore, the number of topics for the data is set to one topic. Considering more topics in the network will give us a chance of having a node to act as publisher and subscriber at the same time. It can be a publisher for a topic and a subscriber for another topic.

The topology of the network in our experiments is a square grid topology. The reason for selecting it is to reduce the simulation time. Other topology options can be considered, such as random topology, where the nodes are distributed randomly within physical terrain boundaries. There is also a cluster topology which is a mix between the grid and the random

topology. In such topology, the physical terrain is divided into some clusters, and the node is placed randomly in each cluster. Also, the developer can define a customized topology scenario.

Another possible direction is to consider the distribution of the base-station functionality over several nodes in the cluster topology especially when the size of the network is significantly large. The cluster head nodes will serve as the cluster base-station and will be responsible for granting access to the nodes within the cluster.

Yet another potential direction is to consider the use of public-key group Elliptic Curve Diffie-Hellman [95] to achieve mutual authentication and establish a shared symmetric key session, and compare it with the devised symmetric key based algorithm implemented in our work.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey. Comput-er Networks, 38, 393-422," ed, 2002.

[2] M. Petrovic, V. Muthusamy, and H.-A. Jacobsen, "Managing Automation Data Flows in Sensor/Actuator Networks," 2007.

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

[4] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Wireless sensor network security: A survey," *Security in distributed, grid, mobile, and pervasive computing*, vol. 1, p. 367, 2007.

[5] A.-H. Jallad and T. Vladimirova, "Data-centricity in wireless sensor networks," in *Guide to Wireless Sensor Networks*, ed: Springer, 2009, pp. 183-204.

[6] B. Krishnamachari, D. Estrin, and S. Wicker, "Modelling data-centric routing in wireless sensor networks," in *IEEE infocom*, 2002, pp. 39-44.

[7] D. A. Tran and L. H. Truong, "Enabling publish/subscribe services in sensor networks," *University of Massachusetts, Boston and IBM Zurich Research Laboratory, Switzerland*, 2010.

[8] P. Boonma and J. Suzuki, "Middleware support for pluggable non-functional properties in wireless sensor networks," in *Services-Part I, 2008. IEEE Congress on*, 2008, pp. 360-367.

[9] C. Esposito, D. Cotroneo, and S. Russo, "On reliability in publish/subscribe services," *Computer Networks*, vol. 57, pp. 1318-1343, 2013.

[10] 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, pp. 305-326, 2008.

[11] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho, and M. Perillo, "Middleware to support sensor network applications," *Network, IEEE*, vol. 18, pp. 6-14, 2004.

[12] S. Hadim and N. Mohamed, "Middleware: Middleware challenges and approaches for wireless sensor networks," *IEEE distributed systems online*, p. 1, 2006.

[13] N. Mohamed and J. Al-Jaroodi, "A survey on service-oriented middleware for wireless sensor networks," *Service Oriented Computing and Applications*, vol. 5, pp. 71-85, 2011.

[14] B. Bhuyan, H. K. D. Sarma, and N. Sarma, "A Survey on Middleware for Wireless Sensor Networks," *Journal of Wireless Networking and Communications*, vol. 4, pp. 7-17, 2014.

[15] C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Sensor information networking architecture and applications," *Personal communications, IEEE*, vol. 8, pp. 52-59, 2001.

[16] P. Bonnet, J. Gehrke, and P. Seshadri, "Towards sensor database systems," in *Mobile Data Management*, 2001, pp. 3-14.

[17] S. Li, S. H. Son, and J. A. Stankovic, "Event detection services using data service middleware in distributed sensor networks," in *Information Processing in Sensor Networks*, 2003, pp. 502-517.
[18] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transactions on database systems (TODS)*, vol. 30, pp. 122-173, 2005.
[19] E. Souto, G. Guimarães, G. Vasconcelos, M. Vieira, N. Rosa, and C. Ferraz, "A message-oriented middleware for sensor networks," in *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, 2004, pp. 127-134.
[20] P. J. Marrón, A. Lachenmann, D. Minder, J. Hahner, R. Sauter, and K. Rothermel, "TinyCubus: a flexible and adaptive framework sensor networks," in *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, 2005, pp. 278-289.
[21] P. Costa, G. Coulson, C. Mascolo, L. Mottola, G. P. Picco, and S. Zachariadis, "Reconfigurable component-based middleware for networked embedded systems," *International Journal of Wireless Information Networks*, vol. 14, pp. 149-162, 2007.
[22] Q. Han and N. Venkatasubramanian, "Autosec: An integrated middleware framework for dynamic service brokering," *IEEE Distributed Systems Online*, vol. 2, pp. 22-31, 2001.
[23] T. Liu and M. Martonosi, "Impala: A middleware system for managing autonomic, parallel sensor systems," in *ACM SIGPLAN Notices*, 2003, pp. 107-118.
[24] P. Levis and D. Culler, "Maté: A tiny virtual machine for sensor networks," in *ACM Sigplan Notices*, 2002, pp. 85-95.
[25] R. Barr, J. C. Bicket, D. S. Dantas, B. Du, T. Kim, B. Zhou, *et al.*, "On the need for system-level support for ad hoc and sensor networks," *ACM SIGOPS Operating Systems Review*, vol. 36, pp. 1-5, 2002.
[26] C.-L. Fok, G.-C. Roman, and C. Lu, "Mobile agent middleware for sensor networks: An application case study," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, 2005, pp. 382-387.
[27] P. Costa, L. Mottola, A. L. Murphy, and G. P. Picco, "Programming wireless sensor networks with the TeenyLime middleware," in *Middleware 2007*, ed: Springer, 2007, pp. 429-449.
[28] C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. L. Murphy, and G. P. Picco, "Tinylime: Bridging mobile and sensor networks through middleware," in *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, 2005, pp. 61-72.
[29] R. de Cassia Acioli Lima, N. S. Rosa, and I. R. L. Marques, "TS-Mid: Middleware for Wireless Sensor Networks Based on Tuple Space," in *Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on*, 2008, pp. 886-891.
[30] G. Banavar, T. Chandra, R. Strom, and D. Sturman, "A case for message oriented middleware," in *Distributed Computing*, ed: Springer, 1999, pp. 1-17.

[31] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys (CSUR)*, vol. 35, pp. 114-131, 2003.
[32] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, *et al.*, "TinyOS: An operating system for sensor networks," *Ambient intelligence*, vol. 35, 2004.
[33] S. L. Keoh, N. Dulay, E. Lupu, K. Twidle, A. E. Schaeffer-Filho, M. Sloman, *et al.*, "Self-managed cell: A middleware for managing body-sensor networks," in *Mobile and Ubiquitous Systems: Networking & Services, 2007. MobiQuitous 2007. Fourth Annual International Conference on*, 2007, pp. 1-5.
[34] L. H. Freitas, K. Bispo, N. S. Rosa, and P. R. Cunha, "SM-Sens: security middleware for wireless sensor networks," in *Information Infrastructure Symposium, 2009. GIIS'09. Global*, 2009, pp. 1-7.
[35] R. Daidone, G. Dini, and M. Tiloca, "STaR: a Reconfigurable and Transparent middleware for WSNs security," in *CONET/UBICITEC*, 2013, pp. 73-88.
[36] P. Chapin and C. Skalka, "SpartanRPC: Secure WSN middleware for cooperating domains," in *Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on*, 2010, pp. 61-70.
[37] D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," in *Acm Sigplan Notices*, 2003, pp. 1-11.
[38] C. Vairo, M. Albano, and S. Chessa, "A secure middleware for wireless sensor networks," in *Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, 2008, p. 59.
[39] S. Marchesani, L. Pomante, M. Pugliese, and F. Santucci, "A Middleware Approach to Provide Security in IEEE 802.15. 4 Wireless Sensor Networks," in *MOBILe Wireless MiddleWARE, Operating Systems and Applications (Mobilware), 2013 International Conference on*, 2013, pp. 85-93.
[40] M. Valero, S. S. Jung, Y. Li, and R. Beyah, *Di-sec: A distributed security framework for heterogeneous wireless sensor networks*: IEEE, 2012.
[41] N. Matthys, C. Huygens, D. Hughes, S. Michiels, and W. Joosen, "A component and policy-based approach for efficient sensor network reconfiguration," in *Policies for Distributed Systems and Networks (POLICY), 2012 IEEE International Symposium on*, 2012, pp. 53-60.
[42] G. Russello, L. Mostarda, and N. Dulay, "A policy-based publish/subscribe middleware for sense-and-react applications," *Journal of Systems and Software*, vol. 84, pp. 638-654, 2011.
[43] G. Dini and I. M. Savino, "A security architecture for reconfigurable networked embedded systems," *International Journal of Wireless Information Networks*, vol. 17, pp. 11-25, 2010.
[44] M. Datasheet, "Crossbow technology inc," *San Jose, California*, 2006.
[45] H. Krawczyk, R. Canetti, and M. Bellare, "HMAC: Keyed-hashing for message authentication," 1997.
[46] S. Blake-Wilson, D. Brown, and P. Lambert, "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)," 2070-1721, 2002.

[47] R. L. Rivest, "The RC5 encryption algorithm," in *Fast Software Encryption*, 1994, pp. 86-96.
[48] *Moteiv Corporation: Ultra low power IEEE 802.15.4 compliant wireless sensor module*. Available: http://www.snm.ethz.ch/snmwiki/pub/uploads/Projects/tmote-sky-datasheet.pdf
[49] *Texas Instruments: Texas instruments cc2420 2.4 ghz ieee 802.15.4 / zigbee ready rf transceiver* Available: http://focus.ti.com/lit/ds/symlink/cc2420.pdf
[50] *U.S. National Security Agency (NSA): SKIPJACK and KEA algorithm specifications (May 1998)*. Available: http://csrc.nist.gov/groups/ST/toolkit/documents/skipjack/skipjack.pdf
[51] D. Eastlake and P. Jones, "US secure hash algorithm 1 (SHA1)," ed: RFC 3174, September, 2001.
[52] V. Rijmen and J. Daemen, "Advanced encryption standard," *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, pp. 19-22, 2001.
[53] P. Chapin. *"Sprocket home page,"*. Available: http://www.cs.uvm.edu/%7Epchapin/Sprocket/
[54] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, 2004, pp. 455-462.
[55] *AVR RZ Raven Datesheet* Available: http://www.atmel.com/Images/doc8117.pdf
[56] D. Wheeler and R. Needham, "TEA extensions (October 1997), Also Correction to XTEA (October 1998)," *Available via: www. ftp. cl. cam. ac. uk/ftp/users/djw3*.
[57] 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*, 2003, pp. 126-137.
[58] G. Bertoni, L. Breveglieri, and M. Venturi, "Power aware design of an elliptic curve coprocessor for 8 bit platforms," in *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*, 2006, pp. 5 pp.-341.
[59] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," in *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, 2004, pp. 71-80.
[60] K. Piotrowski, P. Langendoerfer, and S. Peter, "How public key cryptography influences wireless sensor node lifetime," in *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, 2006, pp. 169-176.
[61] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, 2005, pp. 324-328.
[62] D. Jinwala, D. Patel, and K. Dasgupta, "Optimizing the block cipher and modes of operations overhead at the link layer security framework in the wireless sensor networks," in *International Conference on Information Systems Security*, 2008, pp. 258-272.

[63] Y. W. Law, J. Doumen, and P. Hartel, "Survey and benchmark of block ciphers for wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN),* vol. 2, pp. 65-93, 2006.
[64] J. Lee, K. Kapitanova, and S. H. Son, "The price of security in wireless sensor networks," *Computer Networks,* vol. 54, pp. 2967-2978, 2010.
[65] A. Trad, A. A. Bahattab, and S. B. Othman, "Performance trade-offs of encryption algorithms for Wireless Sensor Networks," in *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*, 2014, pp. 1-6.
[66] M. Çakırolu, C. Bayilmi, T. Özcerit, and Ö. Çetin, "Performance evaluation of scalable encryption algorithm for WSNs," *J. Sci. Res. Essays,* vol. 5, pp. 856-861, 2010.
[67] K. Biswas, V. Muthukkumarasamy, X.-W. Wu, and K. Singh, "Performance evaluation of block ciphers for wireless sensor networks," in *Advanced Computing and Communication Technologies*, ed: Springer, 2016, pp. 443-452.
[68] W. K. Koo, H. Lee, Y. H. Kim, and D. H. Lee, "Implementation and analysis of new lightweight cryptographic algorithm suitable for wireless sensor networks," in *Information Security and Assurance, 2008. ISA 2008. International Conference on*, 2008, pp. 73-76.
[69] K. Biswas, V. Muthukkumarasamy, E. Sithirasenan, and K. Singh, "A simple lightweight encryption scheme for wireless sensor networks," in *International Conference on Distributed Computing and Networking*, 2014, pp. 499-504.
[70] K. Biswas, "Lightweight Security Protocol for Wireless Sensor Networks," in *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on*, 2014, pp. 1-2.
[71] M. Cazorla, K. Marquet, and M. Minier, "Survey and benchmark of lightweight block ciphers for wireless sensor networks," in *Security and Cryptography (SECRYPT), 2013 International Conference on*, 2013, pp. 1-6.
[72] M. Cazorla, S. Gourgeon, K. Marquet, and M. Minier, "Survey and benchmark of lightweight block ciphers for MSP430 16-bit microcontroller," *Security and Communication Networks,* vol. 8, pp. 3564-3579, 2015.
[73] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 162-175.
[74] N. FIPS, "198: The keyed-hash message authentication code (HMAC)," *National Institute of Standards and Technology, Federal Information Processing Standards*, p. 29, 2002.
[75] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway, "UMAC: Fast and secure message authentication," in *Annual International Cryptology Conference*, 1999, pp. 216-233.
[76] T. Krovetz, "Message authentication on 64-bit architectures," in *Selected Areas in Cryptography*, 2006, pp. 327-341.
[77] S. Halevi and H. Krawczyk, "MMH: Software message authentication in the Gbit/second rates," in *International Workshop on Fast Software Encryption*, 1997, pp. 172-189.
[78] A. A. Al-Roubaiey, "Energy-aware publish/subscribe DDS-based middleware for wireless senesor and actuator networks," KFUPM, 2015.

[79] E. Perla, A. Ó. Catháin, R. S. Carbajo, M. Huggard, and C. Mc Goldrick, "PowerTOSSIM z: realistic energy modelling for wireless sensor network environments," in *Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, 2008, pp. 35-42.
[80] E. Yarrkov, "Cryptanalysis of XXTEA," *IACR Cryptology ePrint Archive*, vol. 2010, p. 254, 2010.
[81] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, *et al.*, "PRESENT: An ultra-lightweight block cipher," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2007, pp. 450-466.
[82] J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, *et al.*, "PRINCE–a low-latency block cipher for pervasive computing applications," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2012, pp. 208-225.
[83] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK block ciphers on AVR 8-bit microcontrollers," in *International Workshop on Lightweight Cryptography for Security and Privacy*, 2014, pp. 3-20.
[84] B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Communications magazine*, vol. 32, pp. 33-38, 1994.
[85] P. Syverson, "A taxonomy of replay attacks [cryptographic protocols]," in *Computer Security Foundations Workshop VII, 1994. CSFW 7. Proceedings*, 1994, pp. 187-191.
[86] A. A. Al-Roubaiey, T. R. Sheltami, and A. S. H. Mahmoud, "Id-based routing protocol for wireless network with a grid topology," ed: Google Patents, 2017.
[87] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, 2005, pp. 46-57.
[88] D. R. Raymond, R. C. Marchany, M. I. Brownfield, and S. F. Midkiff, "Effects of denial-of-sleep attacks on wireless sensor network MAC protocols," *IEEE transactions on vehicular technology*, vol. 58, pp. 367-380, 2009.
[89] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad hoc networks*, vol. 1, pp. 293-315, 2003.
[90] J. Deng, R. Han, and S. Mishra, "Defending against path-based DoS attacks in wireless sensor networks," in *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, 2005, pp. 89-96.
[91] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 1-13.
[92] B. Scheers, W. Mees, and B. Lauwens, "Developments on an IEEE 802.15. 4-based wireless sensor network," *Journal of telecommunications and information technology*, pp. 46-53, 2008.

[93] T. Sun, L.-J. Chen, C.-C. Han, G. Yang, and M. Gerla, "Measuring effective capacity of IEEE 802.15. 4 beaconless mode," in *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE*, 2006, pp. 493-498.

[94] A. A. Al-Roubaiey, "Energy-Aware Publish/Subscribe DDS-Based Middleware for Wireless Sensor and Actuator Networks," King Fahd University of Petroleum and Minerals (Saudi Arabia), 2015.

[95] Y. Wang, B. Ramamurthy, and X. Zou, "The performance of elliptic curve based group diffie-hellman protocols for secure group communication over ad hoc networks," in *Communications, 2006. ICC'06. IEEE International Conference on*, 2006, pp. 2243-2248.

Vitae

Name :ABDALLAH HASAN KHALIL RASHED

Nationality :Jordanian

Date of Birth 9/9/1984

Email :abd_rashed2000@yahoo.com

Address :Beit Leed, Tulkarm, Palestine

Academic Background

Ph.D (Computer Science and Engineering)

May 2018

King Fahd University of Petroleum and Minerals

Dhahran, Saudi Arabia.

M.S (Computer Engineering)

December 2012

King Fahd University of Petroleum and Minerals

Dhahran, Saudi Arabia.

B.S (Computer Engineering)

June 2008

An-Najah National University

Nablus, Palestine.