T.C.

ALTINBAŞ UNIVERSITY

Information Technologies

Design Wireless Sensor Network With Software Defined Network

Saif Abdulkareem Jassim

Master Thesis

Supervisor

Dr. Sefer KURNAZ

Istanbul, 2019

Design Wireless Sensor Network With Software Defined Network

by

Saif Abdulkareem jassim

Information Technologies

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2019

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Sefer KURNAZ

Supervisor

Examining Committee Members (first name belongs to the chairperson of the jury and the second name belongs to supervisor)

Asst. Prof. Dr. Zeynep ALTAN	Software Engineering, Istanbul Beykent University	
Asst. Prof. Dr. Sefer KURNAZ	School of Engineering and Natural Science, Altinbaş University	
Prof. Dr. Osman N. UCAN	School of Engineering and Natural Science, Altinbaş University	

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Oğuz ATA

Head of Department

Assoc. Prof. Dr. Oğuz BAYAT

Director

Approval Date of Graduate School of Science and Engineering: ____/____/____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Saif Abdulkareem jassim

DEDICATION

First and foremost, I would like to thank Allah Almighty for giving me the knowledge, ability and opportunity to undertake this research study and to persevere and complete it satisfactorily . Heartfelt thanks goes to my father and my mother. Every success is a direct consequence of their influence in my life and their love. At the end I have to mention my brother and sister for their support and love .

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Supervisor Dr. Sefer KURNAZ for all the knowledge and support he provided during my study for the Master Degree and throughout the work to complete this thesis and I have to mention the kindness and the support to all my friends particularly Saad Badie Younis which did not leave me alone the whole time at the courses and while doing this thesis.

ABSTRACT

Using Software Defined Networking Principles for WSNs

Saif Abdulkareem Jassim

M.Sc., Information Technologies, Altınbaş University

Supervisor: Dr. Sefer KURNAZ

Date : 2019

Pages : 55

Software Defined Networking (SDN) is a telecommunications network architecture that provides the promise of significant improvements in the network performance. SDN can be divided into data plane and control plane. OpenFlow is the most popular implementation of the SDN paradigm. SDN is envisioned as a way to reduce the complexity of network configuration and management.

In this project an architecture based on SDN for wireless sensor networks (WSN) is proposed. A protocol architecture for software defined network and wireless sensor network are introduced.

A new simulator called Cooja and is used for designing the SDN-WISE network. Our WSN network consist of one sink node and twenty-four WSN node. The network is first tested without using SDN controller. Then the WSN network is operated with SDN controller. Both scenarios are evaluated by studding and analysis the main performance parameter (Received packet, Received power, Throughput) and a comparisons is made. The received packet of WSN with SDN controller is increased by (53%) compared by received packet of WSN network without SDN controller. This difference results from the drop of packet. The throughput of WSN network with SDN controller is increased by (53.2%) compared by throughput of WSN network with SDN controller. This difference is mainly due to the drop packets and the use of SDN controller that helps each node to find the best path to the sink node. So, the SDN controller can

minimize the number of dropped packet. The received power of WSN network with SDN controller is increased by (3%) compared by received power of WSN network without SDN. This difference is mainly because two reasons the first one is one of the basic functions in a SDN architecture is the communication between the control and data plane and an increase in the communication will increase the power consumption the second reason is the power consumption increase when more packets need to be received since the packet reception requires to be turned on longer time than ON time.

ÖZET

Yazılım Tanımlı Ağ ile Kablosuz Sensör Ağı Tasarlayın

Saif Abdulkareem Jassim,

Master Derecesi , Bilgi Teknolojileri, Altınbaş Üniversitesi

Danışman: Dr. Sefer KURNAZ

Tarih : 2019

Sayfalar: 55

Yazılım Tanımlı Ağ İletişimi (SDN), ağ performansında önemli gelişmeler vaat eden bir telekomünikasyon ağ mimarisidir. SDN, veri düzlemine ve kontrol düzlemine ayrılabilir. OpenFlow, SDN paradigmasının en popüler uygulamasıdır. SDN, ağ yapılandırmasının ve yönetiminin karmaşıklığını azaltmanın bir yolu olarak öngörülmüştür.

Bu projede, kablosuz sensör ağları (WSN) için SDN'e dayanan bir mimari önerilmiştir. Yazılım tanımlı ağ ve kablosuz sensör ağı için bir protokol mimarisi tanıtıldı.

SDN-WISE ağını tasarlamak için Cooja adlı yeni bir simülatör kullanılır. WSN ağımız bir sink düğümünden ve yirmi dört WSN düğümünden oluşur. Ağ ilk önce SDN kontrol cihazı kullanılmadan test edilir. Ardından, WSN ağı SDN denetleyicisiyle çalıştırılır.Her iki senaryo da ana performans parametresinin (Alınan paket, Alınan güç, Çıktı) incelenmesi ve analizi ile değerlendirilir ve bir karşılaştırma yapılır. SDN denetleyicisine sahip alınan WSN paketi, SDN denetleyicisine sahip olmayan WSN ağı paketine kıyasla (% 53) arttırılmıştır.Bu fark paketin düşmesinden kaynaklanır. SDN denetleyicisi ile WSN ağının çıkışı, SDN denetleyiciyle WSN ağının çıkışı ile karşılaştırıldığında (% 53.2) artar.Bu fark temel olarak damlacık paketlerinden ve her bir düğümün sink düğümüne en iyi yolu bulmasına yardımcı olan SDN denetleyicisinin kullanımından kaynaklanmaktadır. Bu yüzden, SDN kontrol cihazı, bırakılan paket sayısını en aza indirebilir. SDN denetleyicili WSN ağının alınan gücü SDN olmadan WSN ağının alınan gücüyle karşılaştırıldığında (% 3) arttırılır. Bu fark temel olarak, bir diğeri SDN mimarisindeki temel işlevlerden biri olmasının iki nedeni kontrol ve veri düzlemi arasındaki iletişimdir ve iletişimdeki bir artış güç tüketimini artıracaktır.İkinci neden, paket alımının zamanından daha uzun süre açılması gerektiğinden beri daha fazla paket alınması gerektiğinde güç tüketimi artışıdır

Table of Contents

List of content **page no.**

3. DESIGN OF SDN-WISE NETWORK

4. PERFORMANCE ANALYSIS AND RESULTS DISCUSSION

5. CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORKS

List of Abbreviations

Acronym	Definition
SDN	Software Defined Network
WSN	Wireless Sensor Networks
MEMS	Microelectronic mechanical systems
MANET	Mobile ad hoc network
IEEE	Institute of Electrical and Electronics Engineers
LRWPAN	Low rate wireless personal area network
LR-PAN	Low-Rate Personal Area Network
SDWN	Software-Defined Wireless Network
SDN-WISE	**S**oftware Defined Networking-WIreless SEnsor Networks
ICT	Information and Communication Technology
IoT	Internet of Things
ARM	Advanced RISC Machines
TCP	Transmission Control Protocol
MAC	Medium Access Control
VM	Virtual Machines
OS	Operating System
FWD	Forwarding layer
INPP	In-Network Packet Processing
TD	Topology Discovery
RAM	Random Access Memory
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
RPL	Routing Protocol for Low-Power and Lossy Networks
IPv6	Internet Protocol version 6
nesC	Network embedded systems C
HTTP	Hypertext Transfer Protocol

UDP	User Datagram Protocol
MS	Millisecond
dBm	Decibel-milliwatts
GB	Gigabyte
MB	Megabyte

List of Figures

List of tables

1. INTRODUCTION

1.1 HISTORICAL BACKGROUND OF SDN

Software Defined Network (SDN) research and development have attracted the attention of many researchers and companies. The key idea of an SDN is to split the network forwarding function, performed by the data plane, from the network control function, performed by the control plane. This allows a simpler and more flexible network control and management, and also network virtualization OpenFlow is the main SDN implementation [1].

The network controller communicates with OpenFlow switches using the OpenFlow protocol through a secure channel. Using this connection, the controller is able to configure the forwarding tables of the switch. OpenFlow-based SDN technologies enable us to address the high bandwidth, dynamic nature of computer networks; adapt the network functions to different business needs easily; and reduce network operations and management complexity significantly [1].

SDN is also truly an open technology. This leads to greater interoperability, more innovation, and more flexible, cost-effective solutions. If a network is compliant with the right SDN standards, it could be controlled by multiple SDN controller applications. This is better than each network platform having its own management console and commands that increase vendor lock-in and make network management even more complex. Today, multiple SDN standards are evolving in different areas, and successful SDN strategies will always be based on open, interoperable multivendor ecosystems with key open source technologies or standardized protocols [2].

Computer networks are typically built from a large number of network devices such as routers, switches and numerous types of middle boxes (i.e., devices that manipulate traffic for purposes other than packet forwarding, such as a firewall) with many complex protocols implemented on them. Network operators are responsible for configuring policies to respond to a wide range of network events and applications. Network management and performance tuning is quite challenging and thus error-prone [3].

The fact that network devices are usually vertically-integrated black boxes exacerbates the challenge network operators and administrators face. The idea of "Programmable Networks" has been proposed as a way to facilitate network evolution. In particular, Software Defined Networking (SDN) is a new networking paradigm in which the forwarding hardware is decoupled from control decisions. It promises to dramatically simplify network management and enable innovation and evolution [3].

The main idea is to allow software developers to rely on network resources in the same easy manner as they do on storage and computing resources. The field of software defined networking is quite recent, yet growing at a very fast pace [4].

The distributed control and transport network protocols running inside the routers and switches are the key technologies that allow information, in the form of digital packets, to travel around the world. Despite their widespread adoption, traditional IP networks are complex and hard to manage to express the desired high-level network policies, network operators need to configure each individual network device separately using low-level and often vendor-specific commands. In addition to the configuration complexity, network environments have to endure the dynamics of faults and adapt to load changes. Automatic reconfiguration and response mechanisms are virtually non-existent in current IP networks. Enforcing the required policies in such a dynamic environment is therefore highly challenging [5].

1.2 WIRELESS SENSOR NETWORK (WSN)

Wireless Sensor Networks (WSNs) have been widely considered as one of the most important technologies for the twenty first century [6]. Enabled by recent advances in micro**e**lectronic mechanical systems (MEMS) and wireless communication technologies, tiny, cheap, and smart sensors deployed in a physical area and networked through wireless links and the Internet provide unprecedented opportunities for a variety of civilian and military applications, for example, environmental monitoring, battle field surveillance [**7**].

Distinguished from traditional wireless communication networks, for example, cellular systems and mobile ad hoc networks (MANET), WSNs have unique characteristics, for example, denser level of node deployment, higher unreliability of sensor nodes, and severe energy, computation, and storage constraints [8], which present many new challenges in the

development and application of WSNs. In the past decade, WSNs have received tremendous attention from both academia and industry all over the world. A large amount of research activities has been carried out to explore and solve various design and application issues, and significant advances have been made in the development and deployment of WSNs. It is envisioned that in the near future WSNs will be widely used in various civilian and military fields, and revolutionize the way we live, work, and interact with the physical world [9].

1.3NETWORK DESIGN OBJECTIVES FOR SENSOR NETWORKS

The characteristics of sensor networks and requirements of different applications have a decisive impact on the network design objectives in terms of network capabilities and network performance. The main design objectives for sensor networks include the following several aspects [10]:

• **Small Node Size:** Reducing node size is one of the primary design objectives of sensor networks. Sensor nodes are usually deployed in a harsh or hostile environment in large numbers. Reducing node size can facilitate node deployment, and also reduce the cost and power consumption of sensor nodes.

• **Low Node Cost:** Reducing node cost is another primary design objective of sensor networks. Since sensor nodes are usually deployed in a harsh or hostile environment in large numbers and cannot be reused, it is important to reduce the cost of sensor nodes so that the cost of the whole network is reduced.

• **Low Power Consumption:** Reducing power consumption is the most important objective in the design of a sensor network. Since sensor nodes are powered by battery and it is often very difficult or even impossible to change or recharge their batteries, it is crucial to reduce the power consumption of sensor nodes so that the lifetime of the sensor nodes, as well as the whole network is prolonged.

• **Self - Configurability**: In sensor networks, sensor nodes are usually deployed in a region of interest without careful planning and engineering. Once deployed, sensor nodes should be able to autonomously organize themselves into a communication network and reconfigure their connectivity in the event of topology changes and node failures.

• **Scalability:** In sensor networks, the number of sensor nodes may be on the order of tens, hundreds, or thousands. Thus, network protocols designed for sensor networks should be scalable to different network sizes.

1.4 Wireless Sensor Standards

Wireless sensor standards have been developed with the key design requirement for low power consumption. The standard defines the functions and protocols necessary for sensor nodes to interface with a variety of networks. Some of these standards include IEEE 802.15.4, ZigBee, Wireless HART, ISA100.11, IETF 6LoWPAN, IEEE 802.15.3 and Wibree. The following paragraphs describes some of these standards in more detail [11].

- **IEEE 802.15.4:** IEEE 802.15.4 is the proposed standard for low rate wireless personal area networks (LRWPAN's). IEEE 802.15.4 focuses on low cost of deployment, low complexity, and low power consumption. IEEE 802.15.4 is designed for wireless sensor applications that require short range communication to maximize battery life. The standard allows the formation of the star and peer-to-peer topology for communication between network devices. Devices in the star topology communicate with a central controller while in the peer-to-peer topology ad hoc and self-configuring networks can be formed. IEEE 802.15.4 devices are designed to support the physical and data-link layer protocols [12].
- **ZigBee:** It defines the higher layer communication protocols built on the IEEE 802.15.4 standards for LR-PANs. ZigBee is a simple, low cost, and low power wireless communication technology used in embedded applications. ZigBee devices can form mesh networks connecting hundreds to thousands of devices together. ZigBee devices use very little power and can operate on a cell battery for many years. There are three types of ZigBee devices: Zig-Bee coordinator, ZigBee router, and ZigBee end device. Zig-Bee coordinator initiates network formation, stores information, and can bridge networks together. ZigBee routers link groups of devices together and provide multi-hop communication across devices. ZigBee end device consists of the sensors, actuators, and controllers that collects data and communicates only with the router or the coordinator. The ZigBee standard was publicly available as of June 2005 [13].
- **6LoWPAN:** IPv6-based Low Power Wireless Personal Area Networks enables IPv6 packets communication over an IEEE 802.15.4 based network. Low power device can

communicate directly with IP devices using IP based protocols. Using 6LoWPAN, low power devices have all the benefits of IP communication and management.6LoWPAN standard provides an adaptation layer, new packet format, and address management. Because IPv6 packet sizes are much larger than the frame size of IEEE 802.15.4, an adaptation layer is used. The adaptation layer carries out the functionality for header compression. With header compression, smaller packets are created to fit into an IEEE 802.15.4 frame size. Address management mechanism handles the forming of device addresses for communication.6LoWPAN is designed for applications with low data rate devices that requires Internet communication [14].

1.5 Aim of work

The aim work is to:

I. Implement WSN sink node and sensor node on real device (raspberry pi)

II. Study and analyze the software defined networking architecture.

III. Study and analyze the performance analysis of the developed architecture for WSN based on SDN with and without using the SDN controller.

IV. Make a prototype for (SDN_WSN) on real device by running SDN controller on personal computer (pc) and use a numbers of WSN node to connect it to controller ,

1.6 Outline of Thesis

Chapter 1: Introduces the SDN-WISE system then discusses its features and gives the aim of this project.

Chapter 2: Gives theoretical basics for the wireless sensor network and software defined network.

Chapter 3: Presents the design details for the SDN-WISE network.

Chapter 4: Presents the results for the proposed networks.

Chapter 5: Gives the conclusions and suggestions for future works.

2. LITERATURE REVIEW

The software defined networking (SDN) paradigm promises to simplify network configuration and resource management dramatically. Such features are extremely valuable to network operators and therefore, the industrial and the academic research and development communities are paying increasing attention to SDN. Although manufactures of wireless equipment are increasing their involvement in SDN-related activities. Also because of the great number of sensor nodes and high density, sensor node energy, computing power, and storage capacity are limited; network topology changes frequently and has the self-organizing ability. When choosing wireless sensor network routing protocol, we need to take into account the node energy, the communication load balancing, routing protocol fault tolerance, routing protocol security mechanisms, and so on. It is difficult to design a kind that has the most efficient wireless sensor network routing protocol. The SDWN solved some problems such as the energy limit of center node and master node, normal nodes are missing, node zoning and so forth [15].

While it has been a belief for over a decade that Wireless Sensor Networks (WSN) are application-specific, this study is arguing that it can lead to resource underutilization and counter productivity. This study also identifies two other main problems with WSN: rigidity to policy changes and difficulty to manage. It takes a radical, yet backward and peer compatible, approach to tackle these problems inherent to WSN. It proposes a Software-Defined WSN architecture and address key technical challenges for its core component, The sensor OpenFlow. This work represents the first effort that synergizes software defined networking and WSN [16].

2.1 Introduction

This chapter gives an overview of SDN and WSN networks and their architecture. It also explains the software platform.

2.2 Software Defined and Wireless Sensor Network

More and more physical objects are becoming smart and connected using ICT. A key concept for this is Wireless Sensor Networks (WSN), which connects smart objects with each other in a local area using low power wireless communications. WSN is an important building

block for the future Internet of Things (IoT) vision. WSNs may be used for many different types of applications and in many different types of environments, such as in residential homes, apartment buildings, large office buildings, industrial plants, outdoor in urban environments as well as rural areas [17].

Applications may be delay-tolerant collection of sensor readings, responsive smart home applications, tracking of mobile objects or persons, and time- and mission-critical operation of industrial plants, to name a few. For the IoT vision to become true, it is important to keep the costs low and this is mainly achieved through economy of scale in both hardware and software. This means both standardized hardware and software. At the same time, application requirements may force us to tailor every single WSN deployment individually and this can only be achieved by a high degree of reconfiguration capabilities, something that can be realized by using concepts and ideas from Software-Defined Networking (SDN) [18].

The developments in hardware have made reconfiguration possible. The TelosB WSN node platform [19], which is the most commonly used experimentation platform in the research community, is now 10 years old. Today, for example, ARM and its partners have released new energy-efficient microcontrollers, such as the ARM Cortex M-series, which are suitable for WSN nodes. Some of these have a 32-bit architecture and much more program and code memory as well as faster processing capabilities than TelosB. Also the wireless chips have been updated. All this enables much more functions in software [17].

This extra functionality enables a much higher degree of tailoring, but also more advanced in-network processing. The latter also provides shorter communication paths between sensors, actuators, and the data processing. This also leads to more robustness due to less dependence on far away nodes, as well as lower delays and less energy consumption due to the minimized communication paths [17].

2.3 SDN Architecture for WSN

SDN is a concept developed to meet the demands of more flexibility in the networking implementations on Internet routers. In SDN, there is a decoupling defined between the control plane and the data plane with OpenFlow [20] being the currently most successful standard. New

network control and management solutions can easily be deployed by replacing the control plane functionality.

OpenFlow is based on TCP, which means that the control function could run locally on the router or somewhere else, such as at a central server [17].

Through the interface, the controller application can configure the forwarding tables of the router (or network switch). The task of the controller is to provide a coherent image of the entire network and provide it as one single entity towards the SDN applications. Typical SDN applications could be routing, but also other functions, such as access control and software-based traffic analysis, are possible [17]. Figure 2.1 shows the overall architecture of our proposal with the SDN layers indicated. Each WSN node is equipped with a local controller, whose functionality can be as simple as just receiving and executing the commands from the central controller. The central controller communicates over the network with either the used routing protocol if it is running or simple networking principles, such as network-wide flooding [21].

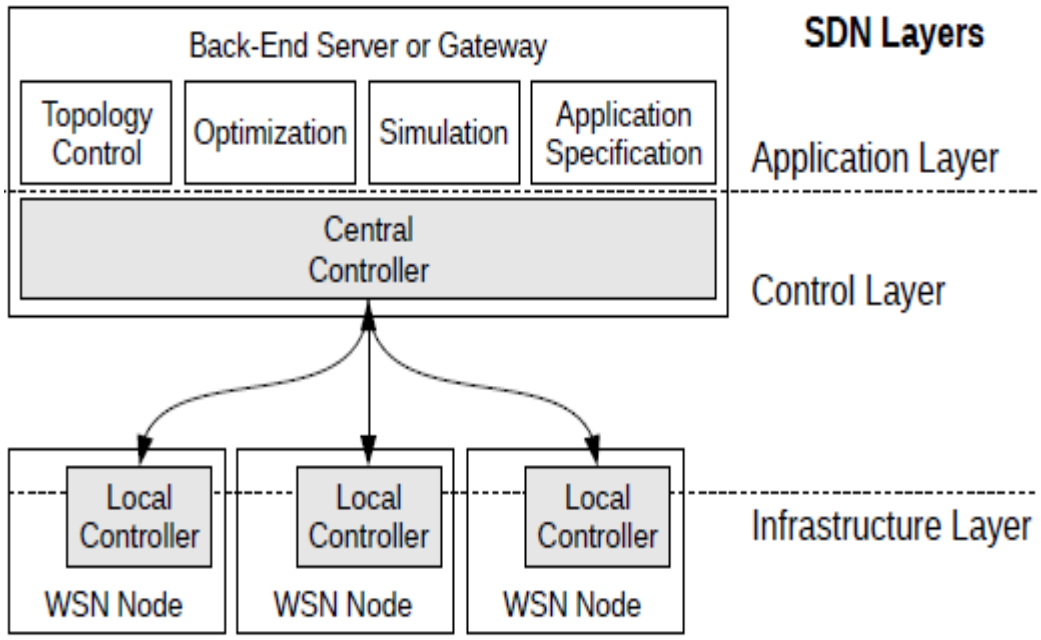

Figure 2.1:Overall SDN architecture for a WSN [17]

On top of this, message formats between the controller and the WSN nodes must be defined. On top of the central controller, there are one or more SDN applications. These applications can be related purely to the networking of the WSN, such as topology control and

routing. In this architecture, some SDN applications may be directed towards the network operator staff with a user interface, while others are completely automated [17].

SDN applications may use optimization solvers, simulators, specifically made algorithms, or a combination. Figure 2.2 shows the functionality on the WSN Nodes. Everything, except potentially some parts of the local controller resides in the infrastructure layer. The main task of the local controller is to setup, reconfigure, and monitor the parts of the software that can be reconfigured. It can do so in two ways. Either it changes parameters in the different functions, such as the central frequency of the radio, the transmission limit in the MAC layer, modifying entries in the forwarding table, etc., or it installs new code in the different functions that changes the behavior. The latter can be done by Virtual Machines (VM), but can also be done with native code and dynamically linked library functions. In this way, not only routing and MAC can be modified by the controller, but there is also flexibility for the neighbor and topology discovery functionality as well as other functions. In-network processing of sensor data is important for the robustness of applications, energy-efficiency, and the reduction of the delay [17].

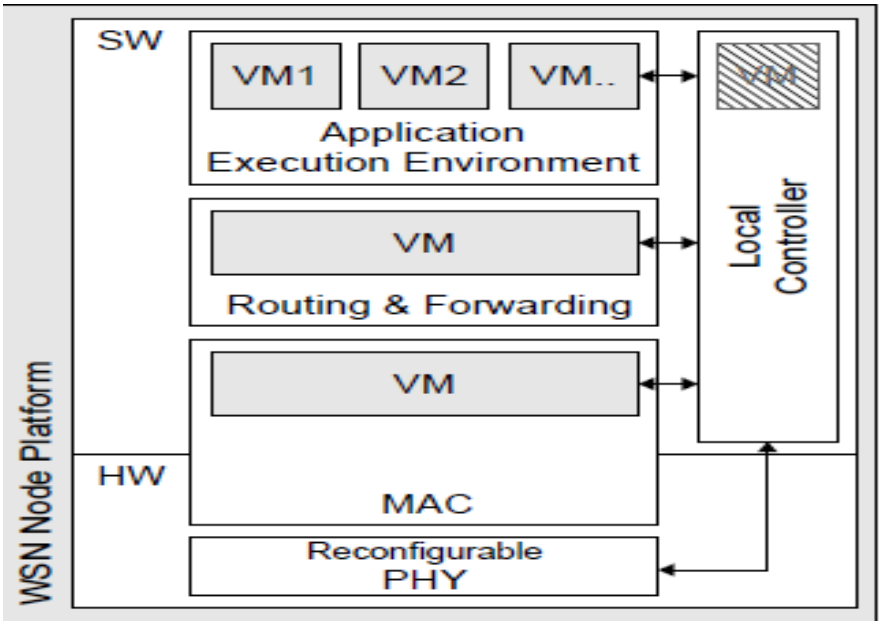

Figure 2.2: The architecture of the WSN node with SDN support [17]

An application execution environment is defined that allows application code to be updated over the air that can process sensor data as it is being forwarded by the nodes. For

instance, the Pro FuN Task Graph Tool [22] can be used. With a good code execution environment in place on the WSN nodes, it also becomes possible to distribute the controller function. I.e., it is no longer required to run the controller on a central node. Furthermore, hybrid controller versions can be defined. In homogenous networks, native binaries and dynamic linking can be used, while massively heterogeneous networks may benefit from using byte code and VM technology designed for embedded systems. Contiki OS has good support for dynamic loading as well as over the air programming, which can be used if all run the same software and micro-controller architecture. If various hardware and software systems exist in the same network, VMs may be a better choice. Some researchers have proposed to put a virtual machine (VM) on WSN nodes. In this way, one single byte code binary can be distributed and executed on any node. Examples of VMs developed for WSN nodes include EmbedVM, TakaTuka [23].The latter two are Java VMs. A third option is using something similar to the command chains of snapMac [24], i.e., a domain-specific VM-like engine with a small set of pre-defined commands that can be executed by a very simple byte code interpreter. A fourth option is to use very simple scripting languages. Whatever choice is made, it is also important to have a good run-time monitoring system that, for instance, can deal with application processes running out of hand or debugging. VMs and scripting languages may be better in offering that functionality [17].

2.4 Software Defined Networking for Wireless Sensor Networks

Software Defined Networking (SDN) is envisioned as a way to reduce the complexity of network configuration and management. New network control and management solutions can be easily deployed on existing equipment as simply as it is to install new programs on a computer. In SDN networks, management operations are centralized and physically separated from forwarding operations. In fact, SDN Switches classify and forward packets according to the so called *flow rules* sent by the SDN Controller(s). When a switch has no information available to classify an incoming packet, it requests the assistance from one or several controllers which should provide an appropriate rule. The policies applied by controllers, which in the end define the entire network behavior, can be easily and rapidly modified by installing a new Controller software. SDN-WISE uses this paradigm in wireless infrastructure less networks [25].

2.5 SDN-WISE Protocol Architecture

The protocol architecture of SDN-WISE is shown in figure 2.3.

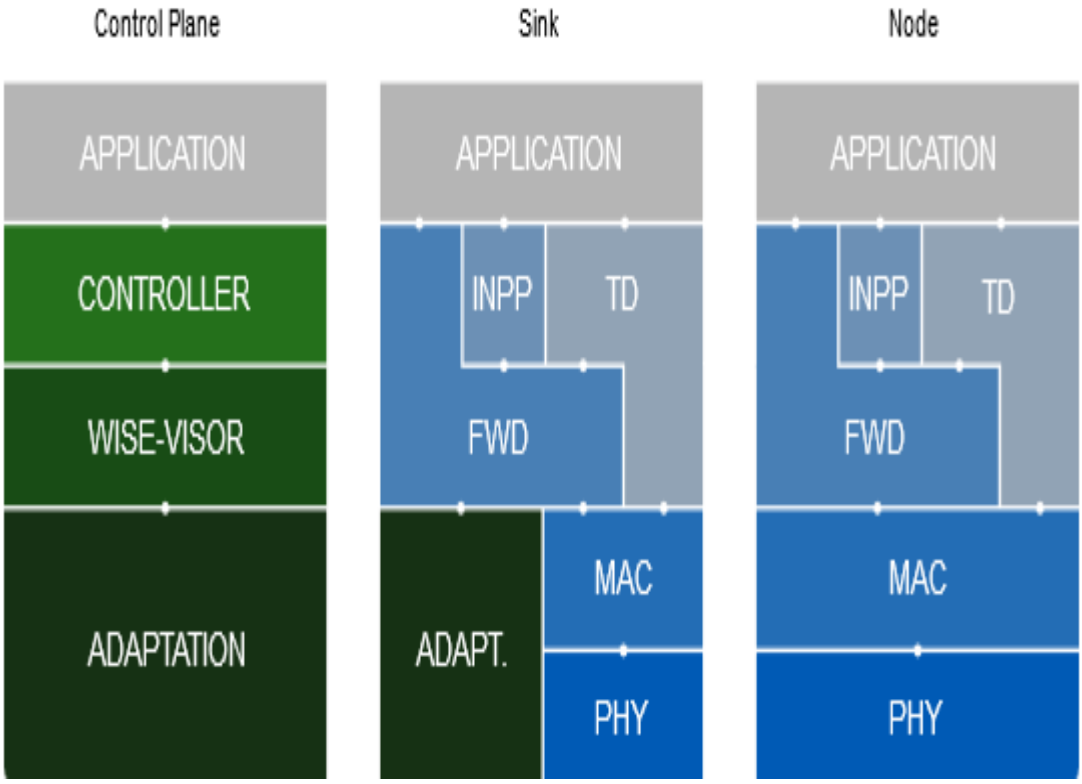

Figure 2.3: protocol architecture of SDN-WISE [25]

SDN-WISE is based on IEEE 802.15.4 physical and MAC layers. Network elements can be distinguished into Sinks and Nodes. The difference between Sinks and the Nodes is that the formers are equipped with a network interface connected to an infrastructure network. Therefore, all control packets should find their way towards a Sink to leave the WSN and reach the Controller.

On top of the MAC layer, the Forwarding (FWD) layer handles incoming packets as specified in the WISE Flow Table. The FWD layer updates this table according to the configurations sent by the Control plane [25].

The In-Network Packet Processing (INPP) layer run on top of the Forwarding layer and it is responsible for operations like data aggregation or other in-network processing. If no entry in the WISE Flow Table matches the current packet, a request is sent to the Control plane. In order

to contact the Control plane, each node has to know its best next hop towards a Sink. This value is calculated in a distributed way using the Topology Discovery (TD) layer through beaconing. In the Control Plane, network logics are dictated by one or several Controller(s), and a WISE-Visor. The WISE-Visor abstracts network resources so that different logical networks, with different management policies set by different Controllers, can run over the same set of physical devices.

Between the Sinks and the WISE-Visor there is the Adaptation layer which is responsible for formatting messages received from a Sink in such a way that they can be handled by the WISE-Visor and vice versa [25].

2.5.1 Proposed Architecture: Control Plane

Controller: It controls network topology representation, routing and other parameters management

Wise-visor: It Abstracts the network resources so different logical networks can be created.

Adaptation: It formats packets received from the WSN (Real/Simulated) in such a way they can be handled by the upper layers and vice versa.

2.5.2 Proposed Architecture: Data Plane

Topology Discovery (TD): It manages Parameters Configuration, Beaconing, Neighbor's reporting.

In-Network Packet Processing (Inpp): It Is responsible for Data aggregation, *Network Coding* Compressive Sensing.

Forwarding (Fwd): It handles incoming packets as specified in the WISE Flow Table.

3. DESIGN OF SDN-WISE NETWORK

3.1 Introduction

This chapter presents the design details for the WSN network with and without using SDN controller and implement sdn-wise network in real node.

3.2 Contiki

Contiki is a relatively new open source operating system which “allows tiny, battery-operated low-power systems to communicate with the Internet” [26]. Contiki was built primarily by a researcher from the Swedish Institute of Computer Science, Adam Dunkels. While still heavily involved with the project, the open source community has embraced his work and continued to build upon his contributions. Contiki is special in that it is designed specifically to accommodate for the limiting factors of motes in wireless sensor networks. By present-day standards, the computational power of a mote is substantially less than that of a personal 6 computers. For example, the average mote is outfitted with an 8-bit microcontroller and tens of kilobytes of RAM. It is with that in mind that Contiki was built to be incredibly lightweight [27].

While initially not very popular, it is now starting to gain some traction in the wireless sensor networking world due to the impressive list of features it has over its competitor, like TinyOS. The fully-compliant IPv6 stack found in Contiki was certified (and contributed to) by Cisco [26]. In addition to support for IPv6, Contiki also brings newer, low-power protocols like 6lowpan, RPL, and CoAP to its users. If a wireless sensor implementation does not need the full IPv6 stack, users can elect to use Contiki’s lightweight Rime networking stack. This will break compatibility with the traditional Internet, but can reduce resource utilization on the mote in appropriate scenarios [28].

3.3 Differences from TinyOS

Up until a few years ago, TinyOS was the only wireless sensor operating system available. Originally developed by a team of researchers from the University of California at Berkeley, it is credited as being the first operating system specifically for wireless sensor networks. Above all else, it strives to achieve the tiniest memory footprint possible. One of the ways in which the

developers achieved this was by electing to code the operating system in nesC, a dialect of the C programming language [28].

NesC is intended for use in small, embedded networking devices and thus seems like a logical choice for a wireless sensor network operating system. Among the modifications to C that can be found in nesC is the concept of split-phase operations. Split-phase operations aim to eliminate blocking-wait calls which can seize control of the processor and waste valuable CPU time, and thus power. Unlike traditional C, nesC divides methods at the point where they invoke a lengthy system call. Due to this segmentation, the developer now becomes responsible for writing callback methods which are executed upon the completion of these system calls which are often unpredictable in duration. For example, if the developer wishes to turn the wireless radio on, they invoke the *on* method for that radio and then write the next part of their program logic in the radio's on complete callback method. This callback method is automatically invoked once the hardware has reported that the wireless radio has either been powered on or an error occurred while attempting to do so [28].

Contiki abandons nesC in favor of the standard C programming language and GNU development tools. nesC, and thus TinyOS, require a modified set of tools in order to build the operating system and user programs. This is just an added nuisance to the developer who now must maintain a separate development environment when dealing with TinyOS. It is for that reason that TinyOS is, by default, packaged and distributed as a virtual machine appliance. Of course, a developer may elect to install the necessary nesC components themselves but this is actually strongly discouraged by members of the TinyOS community. For the sake of convenience Contiki is also released as a virtual machine variant, dubbed Instant Contiki [28].

3.4 Cooja: The Contiki Simulator

One of the major advantages to the Contiki operating system for sensor software developers is the inclusion of the Cooja simulator environment. Cooja is a feature-rich simulator which allows developers to virtually deploy an environment of motes and gather a host of metrics about each one. Additionally, Cooja is capable of emulating several hardware platforms that Contiki could be compiled for. Better yet, it can also simulate a wireless radio medium, enabling developers to debug their applications and behave how they would function in a real-world scenario.

Perhaps most importantly for a project such as this, it is bundled with Instant Contiki and is completely free of charge. For these reasons, all debugging and experimentation for this project occurred in the Cooja simulator.

Cooja enables the user to have control over a few parameters for the simulation environment. To account for interference and packet loss, one can modify the quality of the radio medium and even adjust, on a mote-by-mote basis, the probability with which that mote will successfully send or receive a packet. Cooja also allows for a maximum mote startup delay to be set along with a random seed. These parameters are used to calculate a random amount of time that it will take each mote to complete its boot sequence and begin executing its program. This accounts for the time that would be taken to connect each mote to a source of power. If all the motes are being powered by AA batteries, the odds of them all being inserted by a human being and booting at the same exact instant are slim.

To ease the development of some operating system components, like WPI-MAC, this can be disabled so that all motes begin executing at the same exact moment in time. The settings used for the development and testing of WPI-MAC are explained in greater detail later in this report.

3.5 Wireshark

Wireshark is an open source network packet analyzer tool that captures data packets flowing over the wire (network) and presents them in an understandable form.
Some of the important benefits of working with Wireshark are:

- **Multiple protocol support**: Wireshark supports a wide range of protocols ranging from TCP, UDP, and HTTP to advanced protocols such as AppleTalk.
- **User friendly interface**: Wireshark has an interactive graphical interface that helps in analyzing the packet capture. It also has several advance options such as filtering the packets, exporting packets, and name resolution.
- **Live traffic analysis**: Wireshark can capture live data flowing on the wire and quickly generate information about its protocols, flow media, communication channels, and so on.
- **Open source project**: Wireshark is an open source project and most of its development has been carried out through contribution from over 500 developers around the globe. We can write our own code and add to the project to meet our specific requirements.

3.6 Design and Implementation of Proposed Network Using Cooja Simulator

The network was designed and implemented using the COOJA Simulator. The project consists of two scenarios. In the first scenario, the network consists of one sink node and twenty-four WSN node. While in the second scenario, the network has one sink node, twenty-four WSN nodes and SDN controller. Table 3.1 shows the WSN network parameters.

Table (3.1) WSN network parameters

Parameter name	Value
Network scale	X=Y=100 meter
Coverage area of nodes	40 meter
Speed limit	100%

3.6.1 Fist Scenario: WSN Network Without SDN Controller

This scenario illustrates the design of WSN network without using SDN controller. The following figures shows the design steps of the network. As shown in figure 3.1, the login page consist of password login bar that must be filled with a word "user" as password.

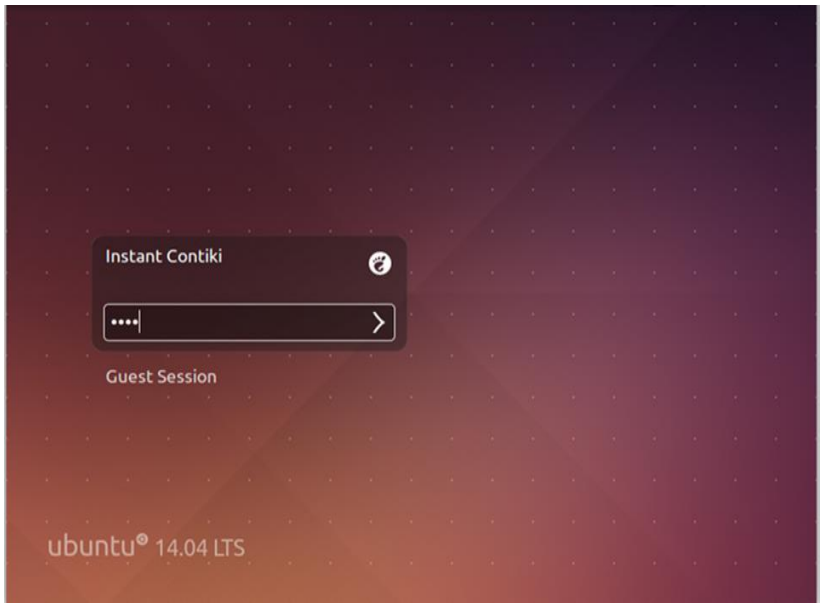

Figure 3.1 Cooja login interface

After starting Cooja, a blue empty window appeared as shown in figure 3.2. This window contains file, simulation, mote, tools, settings and help options in the taskbar each of which has its specific features that helps in the design process.

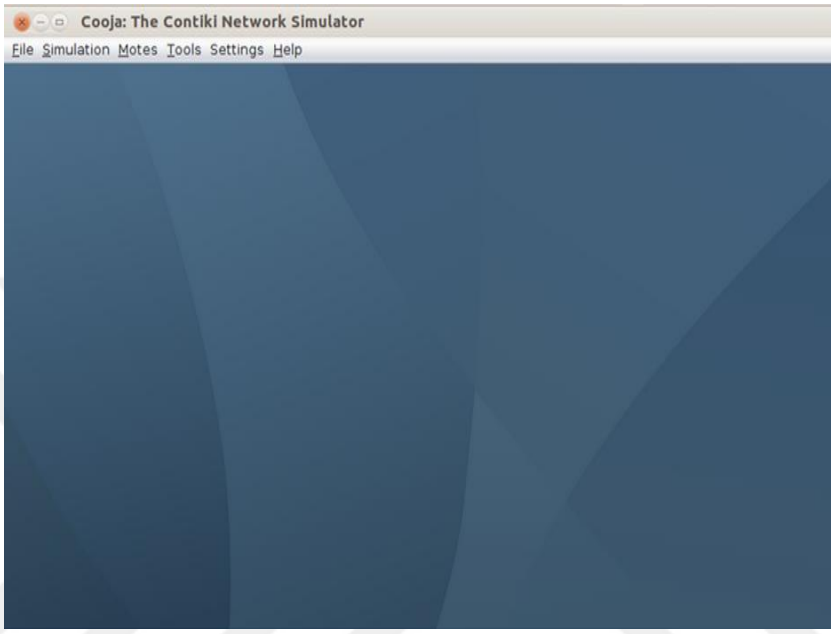

Figure 3.2: Cooja interface

To create a virtual WSN network, we had created a new simulation. As shown in figure 3.3

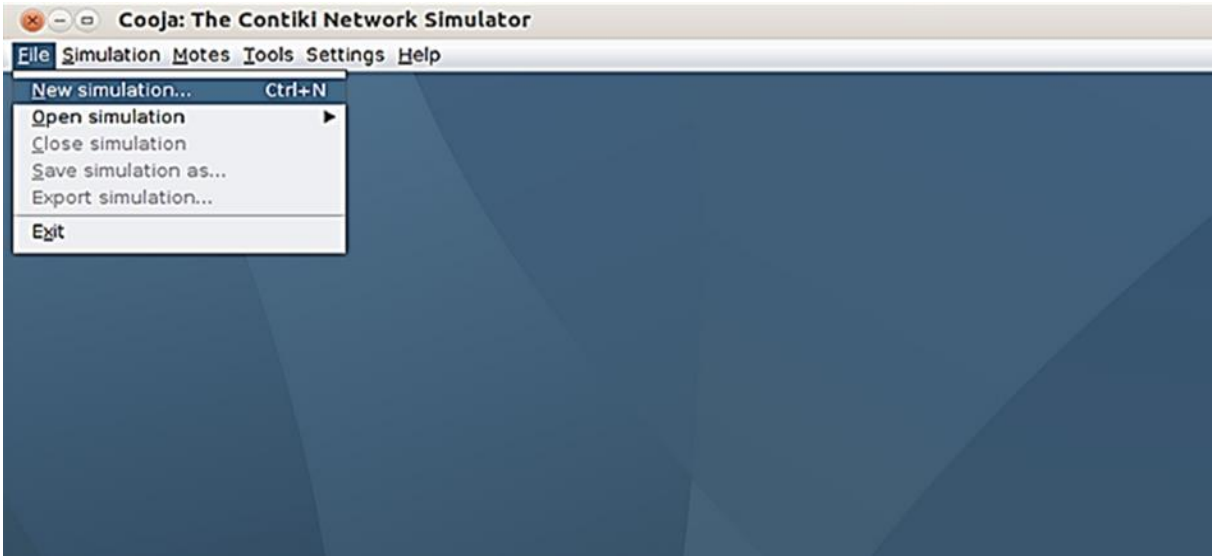

Figure 3.3: Create new simulation

After the creation of the simulation platform, the simulation features had been entered as illustrated in figure 3.4, where the simulation name was SDN-WISE, the random seed equals 123.456 and the Mote startup delay equals 1.000 ms.

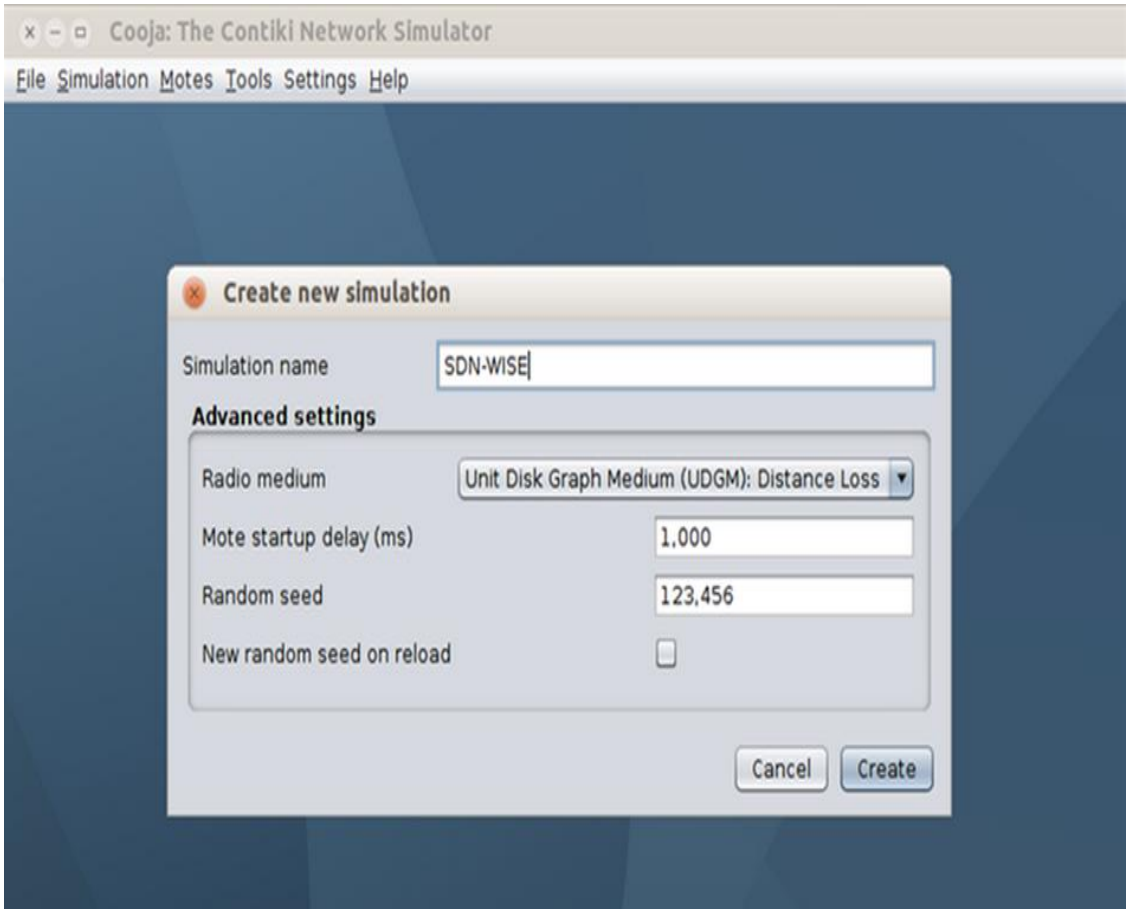

Figure 3.4: Setup new simulation

After the creation of the simulation, Cooja brings up the new simulation as show in figure 3.5. The Network window, at the top left of the screen shows all the motes in the simulated network - it was empty, since we have no motes in our simulation. The Timeline window, at the bottom of the screen, shows all communication events in the simulation over time, which is very handy for understanding what goes on in the network. The Mote output window, on the right side of the screen, shows all serial port printouts from all the motes. The Notes window on the top right is where we can put notes for our simulation. And the Simulation control window was where we had started, pause, and reload our simulation.

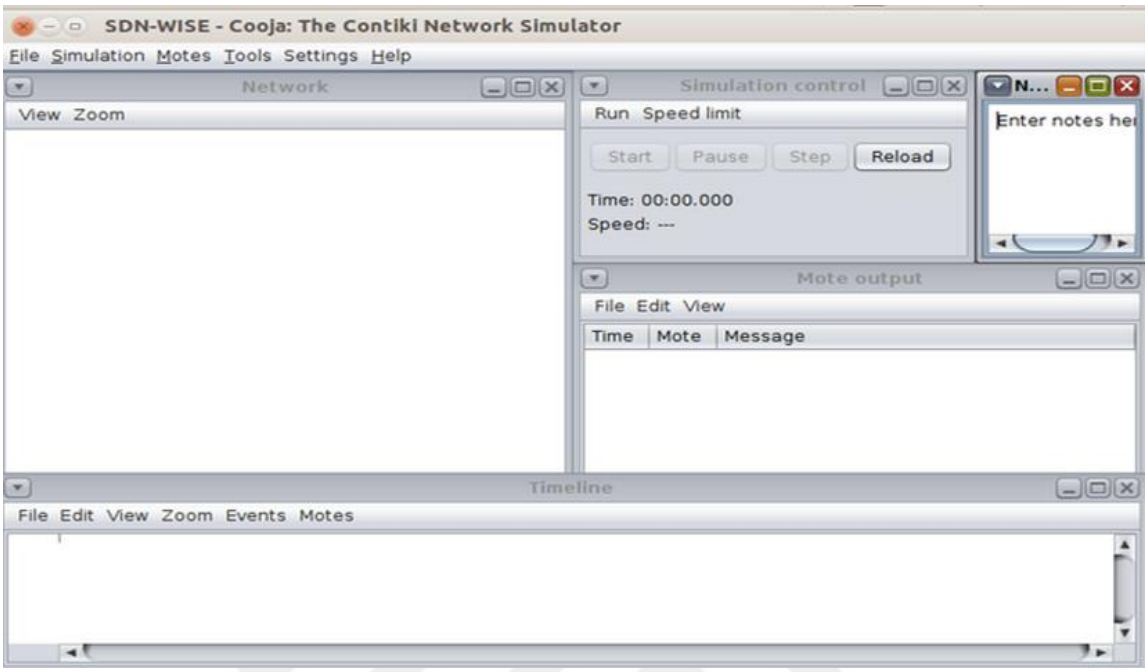

Figure 3.5: New simulation interface

After that, a Sink was added to the simulation by clicking on mote then adding a mote which leads to the creation of new mote and finally import java mote.

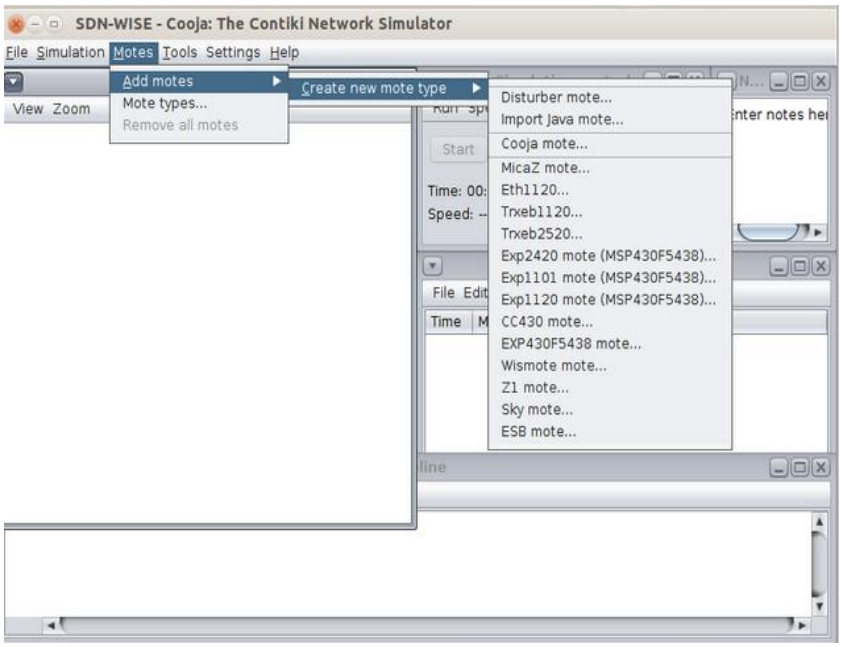

Figure 3.6: Add sink node

In the Creation of Mote Type: Application Mote was first selected followed by the Browse button and then Selecting Class as shown in figure 3.7.

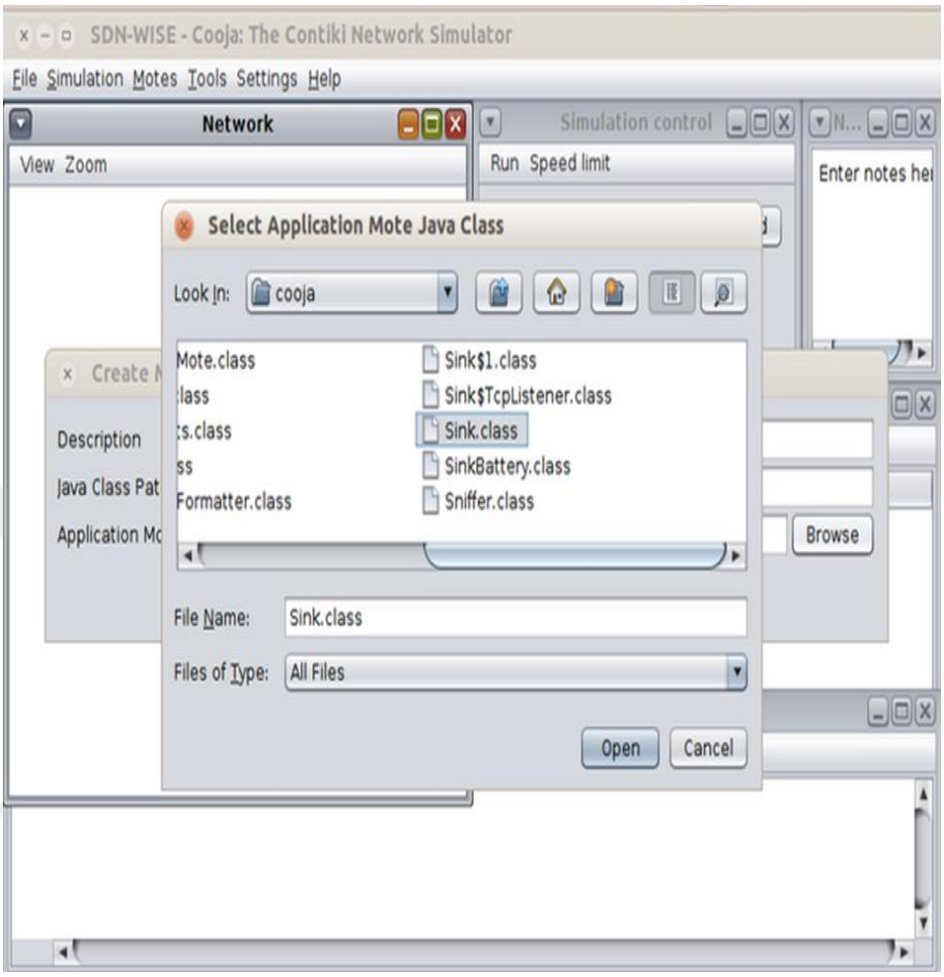

Figure 3.7: Select sink node

In the description field, SDN-WISE sink was written, the create button was selected and then the Add motes button was chosen as shown in figures 3.8 & 3.9:

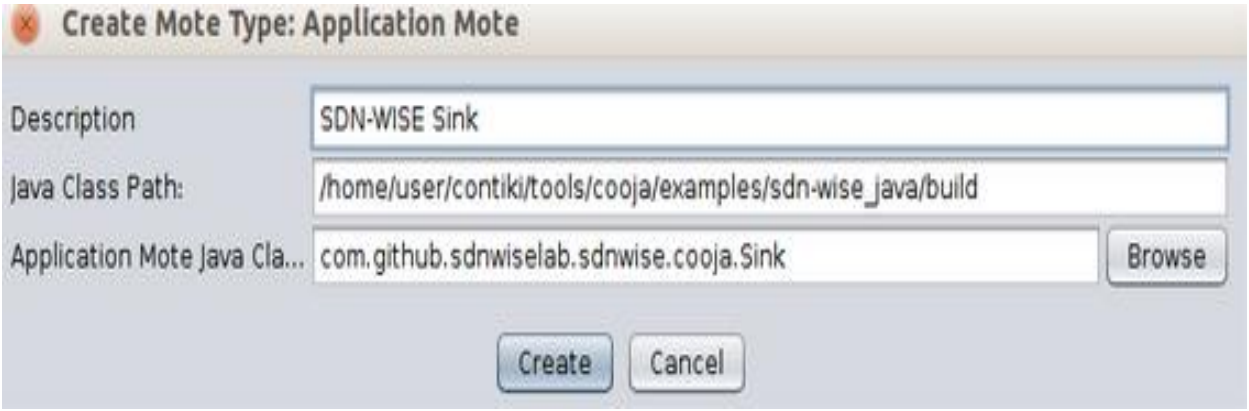

Figure 3.8: Description of sink node

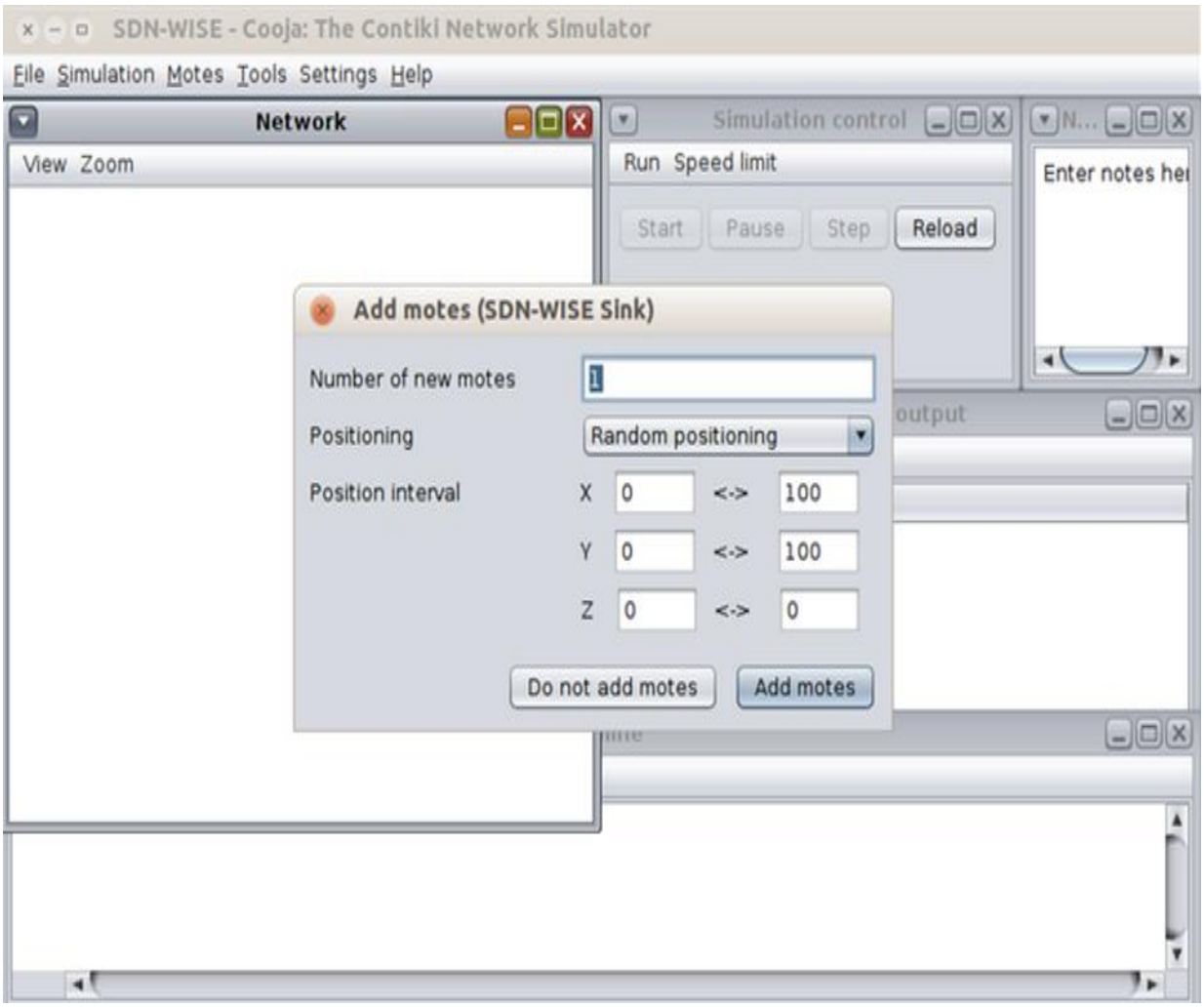

Figure 3.9: Set parameter of sink node

The last step was setting the X, Y and Z parameters of the sink node and both X and Y were set to 100 meter and the position of network was set to random. The previous steps were repeated but by selecting the file Mote. Class instate of Sink. Class.

Figure 3.10 shows the WSN network and use of the mote output window to display the output of sink node. When the view button is clicked, another window will be display that allows you to choose the mote output or radio traffic.

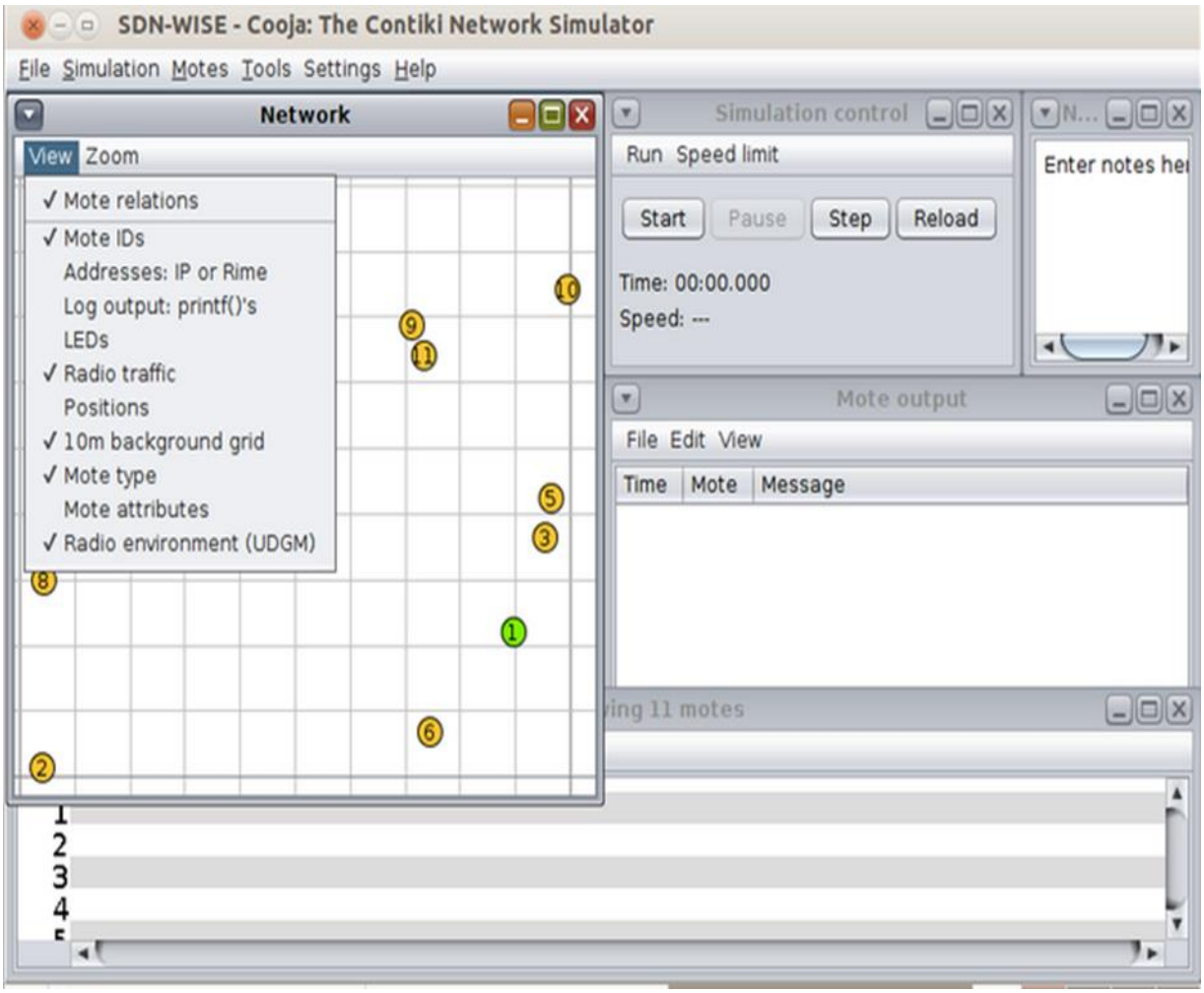

Figure 3.10: View option of network

All the nodes can be reached by the sink, directly or by multiple hops, as shown in Figure 3.11, that can be seen by clicking on a node. Then, a green circle appeared. All the nodes inside the green area can be reached by the selected one. Also a gray circle was there which represents the interference area of the node which was set to zero.

After that the WSN network was run, so the radio traffic can be seen between the sink node and the other node and the number of each node and coverage area can also be seen. This operation can be shown briefly in figure 3.11.

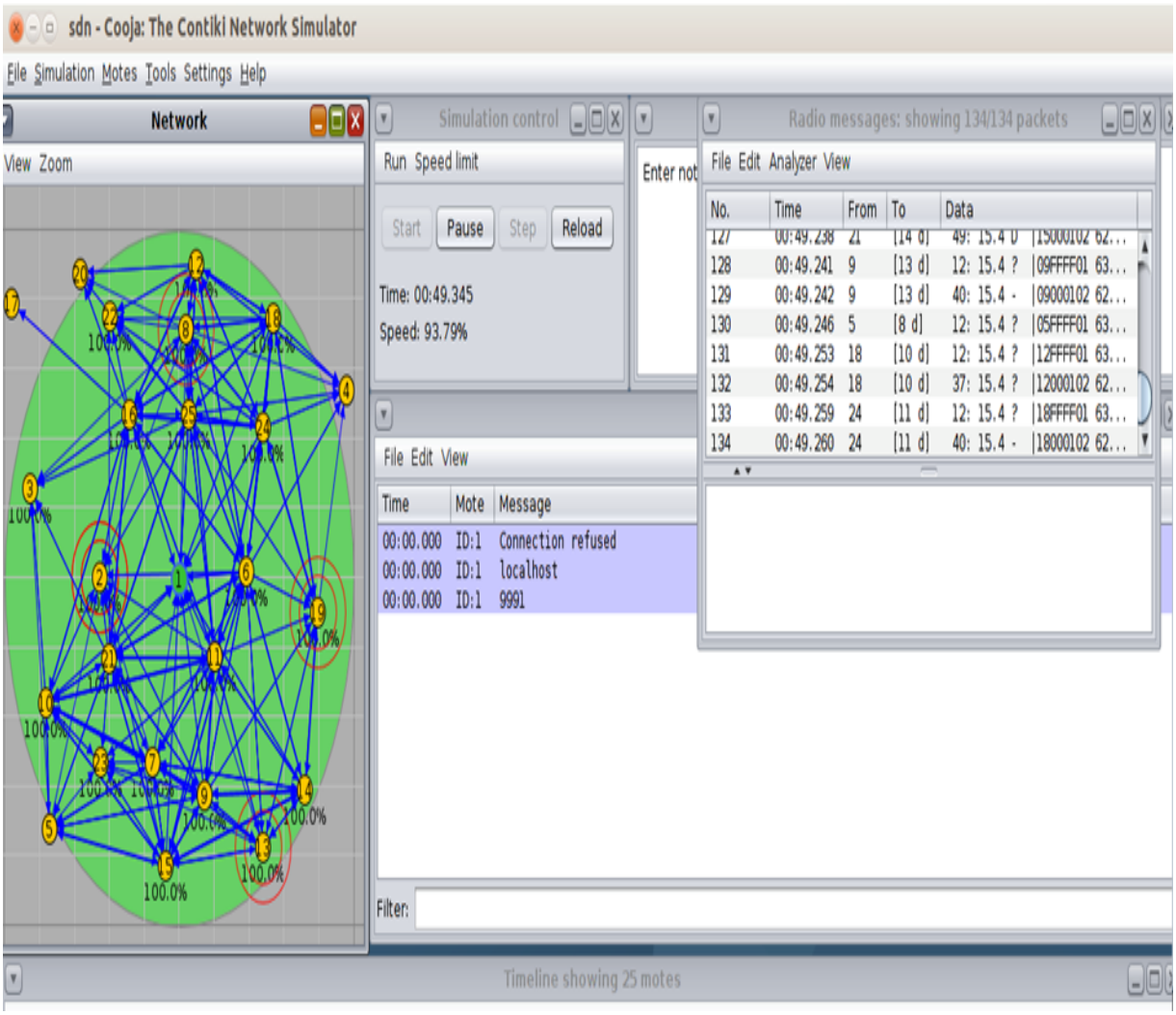

Figure 3.11: Overall WSN network

3.6.2 Second Scenario: Wireless Sensor Network with SDN Controller

In this scenario SDN controller was used with WSN network. The controller uses Dijkstra's routing algorithm. The controller responds to the requests coming from the nodes and after 60 seconds it sends messages containing the string "Hello World!" to the nodes. The sink node is the first node in the network receiving the messages sent by the user. It checks in its WISE Flow Table, if it does not find a matching rule then it asks the controller about it. The controller provides a path to reach the destination and sends the corresponding rules. The nodes in this path learn the rules and thus can correctly forward the message.

When a node receives a message for itself it prints the payload in the Mote output window

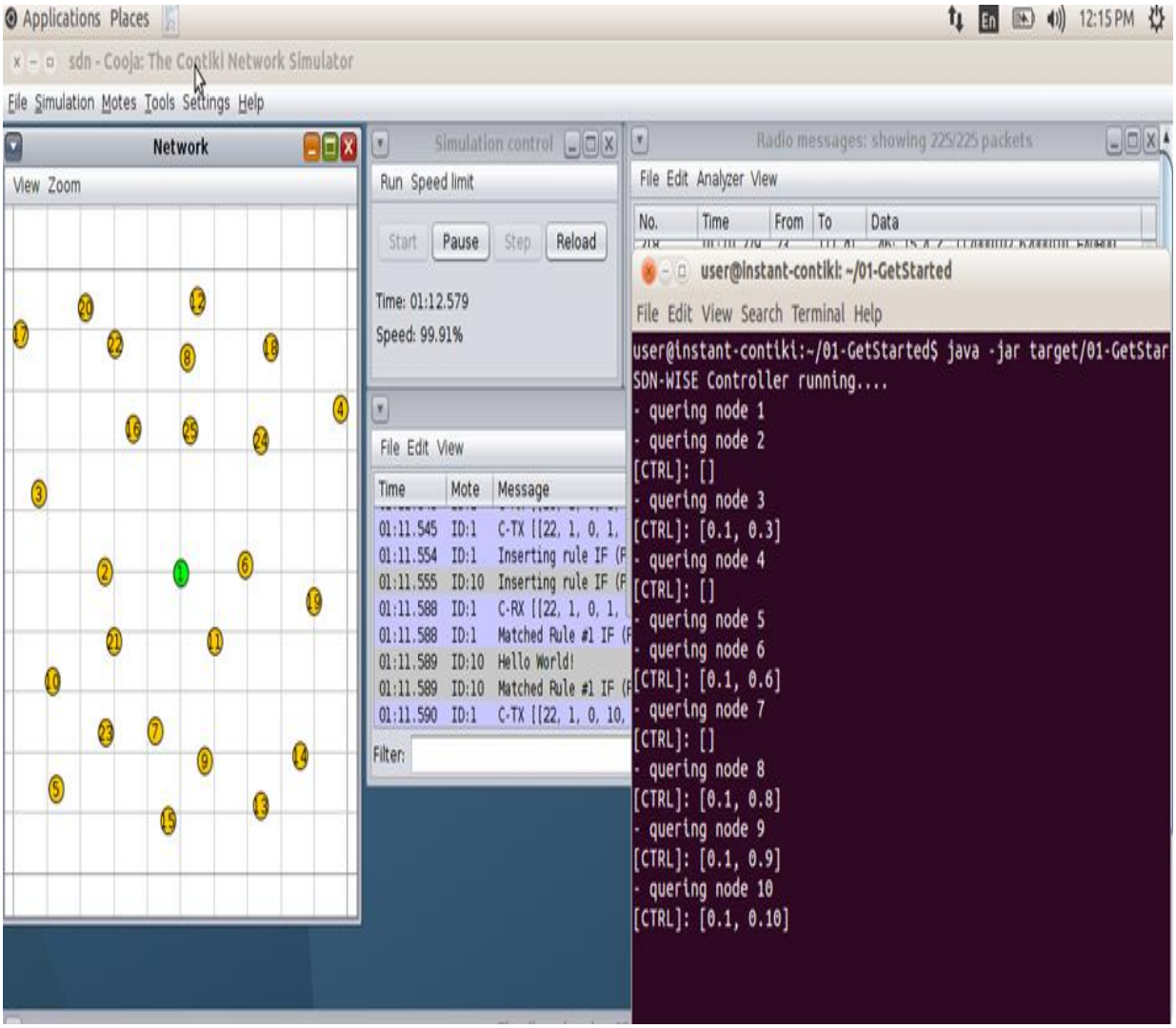

Figure 3.12: SDN controller with WSN network

Then, the trace file of each scenario was saved with extension pcap. The pcap file was used in Wireshark program to evaluate the performance of network before using SDN controller and after using SDN controller then the results were compared.

3.7 Hardware implementation

In our testbed we used ZI motes which is developed by Zolertia. Z1 is low power wireless module, The module provide IEEE 802.15.4.Usually Z1 mote used for testing Wireless sensor network. Each mote equipped with 8 kB RAM and 92 kB Flash Memory, and its CC2420 transceiver has an effective data rate of 250 kbps and operates at 2.4 GHz. Z1 can battery powered or USB powr by connect it to one of wireless sensor network open source operating system like Contiki OS by using USB cable by using their micro usb.

Figure 3.13: Z1 Mote

For both sink node and sensor node we use SDN-WISE source code which is integrate with contiki operating system. one sensor node will be installed as sink node and rest sensor node will be installed as normal node.

In order to upload source code for sink node and sensor node to Z1 mote we must first use set of toolchain. The following toolchain dependencies has been installed in our system (Linux-Ubuntu 32-bit) :

```
sudo apt-get update
sudo apt-get install gcc-arm-none-eabi gdb-arm-none-eabi
sudo apt-get install build-essential automake gettext
sudo apt-get install curl graphviz unzip wget
sudo apt-get install gcc gcc-msp430
```

Figure 3.14: Toolchain dependencies for sink node and sensor node

Contiki operating system dose not run any application or project without Make file so that project or application compiled correctly.

In our thesis SDN-WISE use the following Makefile which can be edit to any specification you need :

```
CONTIKI_PROJECT = sdn-wise
all: $(CONTIKI_PROJECT)
SMALL = 1
CONTIKI = ../contiki
CONTIKI_WITH_RIME = 1
PROJECT_SOURCEFILES = flowtable.c packet-buffer.c address.c
    neighbor-table.c
    packet-handler.c node-conf.c packet-creator.c
TARGETDIRS += ../targets
CFLAGS += -DPROJECT_CONF_H=\"project-conf.h\"
include $(CONTIKI)/Makefile.include
```

Figure 3.15: Makefile of SDN_WISE

```
#!/bin/bash
TARGET="z1"
make TARGET=$TARGET DEFINE=SINK=1
mv sdn-wise.$TARGET compiled/sink.$TARGET
make TARGET=$TARGET DEFINE=SINK=0
mv sdn-wise.$TARGET compiled/node.$TARGET
```

Figure 3.16: bash Algorithm for SDN_WISE

Bash script on figure () which is provided with SDN-WISE source code .Was used to compiled the source code of SDN_WISE code After the source code is compiled . The source code was uploaded to Z1 mote. We Uploaded the source code to mote by connect Z1 mote to Computer by using mini-USB port.

Figure ( ) show the Code that used to upload the source code of SDN_WISE to Z1 mote .

```
sudo make compiled/node.upload MOTES=/dev/ttyUSB0
```

Figure 3.17: Command for uploading code to z1 mote

In order to input data or to display output of nodes this done by connect it to serial port by execute these one of the following command shown in fig ()

```
sudo make login MOTES=/dev/ttyUSB0
sudo make serialdump MOTES=/dev/ttyUSB0
```

Figure 3.18: Input data or display data after connect mote to serial port

After that we can use simulations modeling the behavior of the sensor nodes and the sinks can be executed on PC. For our work we use Cooja to emulate modeling Node 0 is the sink and interacts through the Adaptation module with a real instance of the WISE-Visor. Accordingly. The (emulated) Sink can be used to create a virtual network extension so that simulated and real nodes are fully integrated and can interact with each others. This can be useful for testing a real

network scenario in which there are not enough real devices. In this case only one Controller is used for both nodes.

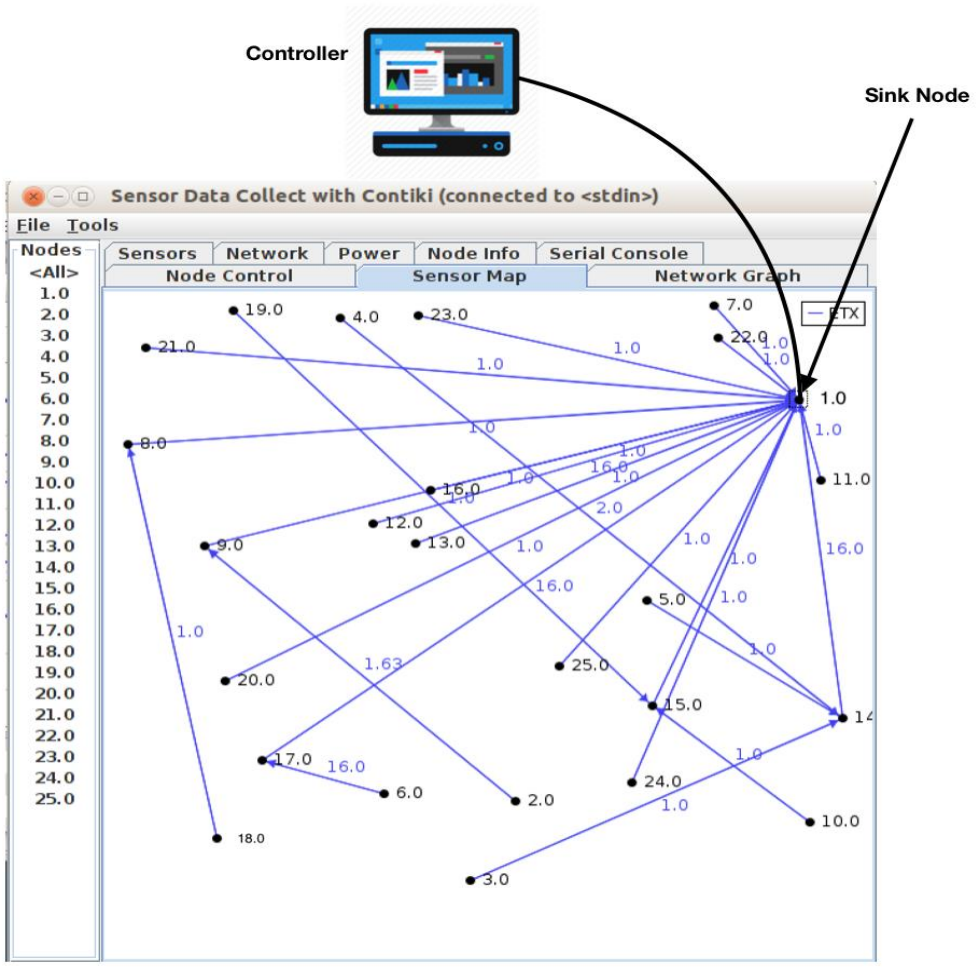

Figure 3.19: show Integration of network with the Cooja simulator.

4. Performance Analysis and Results Discussion

4.1 Results Overview

This chapter will present and discuss results about simulations of the proposed WSN network. Also, shows the comparison between WSN network with SDN controller and without SDN controller. The overall time of the simulation for each scenario takes 4.7 minutes. The results show WSN and WSN-SDN parameters (Throughput, Received packet). For this project we use laptop operating system mac (64-bit) , RAM is 8 GB and dual core intel core m .For the virtual machine we use contiki operating (32-bit),actual size is 8.89 GB and base memory is 5659MB.

4.2 Results of Proposed Network Using Cooja Simulator and Wireshark Program

4.2.1 Network Received Packet

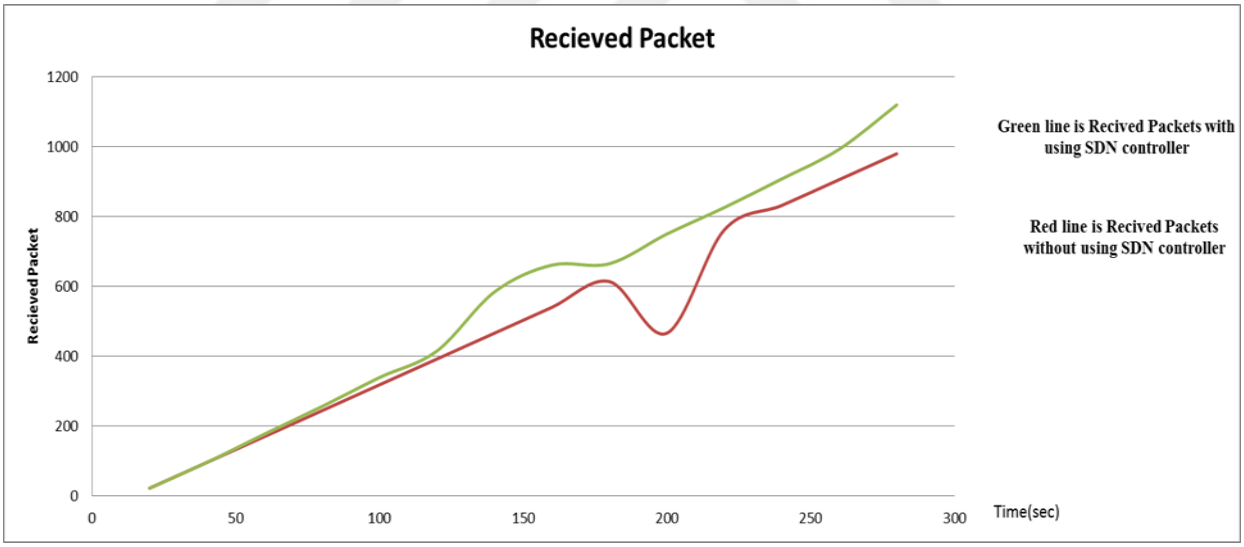

Figure 4.1: Received packet of WSN network with and without the SDN controller

It is clear from figure 4.1 that the received packet of WSN without SDN controller is less than the received packet of WSN network with SDN controller and in some point the difference between the two scenarios is very high. This difference results from the drop of packet.

4.2.2 Network Throughput

The throughput mean amount of data transferred from one place to another. Figure 4.2 shows the throughput of WSN network with and without the SDN controller. It is clear from figure 4.2 that the throughput of WSN network without SDN controller is less than the throughput of WSN network with SDN controller. This difference is mainly due to the drop packets and the use of SDN controller that helps each node to find the best path to the sink node. So, the SDN controller can minimize the number of dropped packet.

The throughput formula [30]

$$Throughput = \frac{Protocol-Window-Size-in-bits}{Time-in-seconds} = Bits-per-second-throughput$$

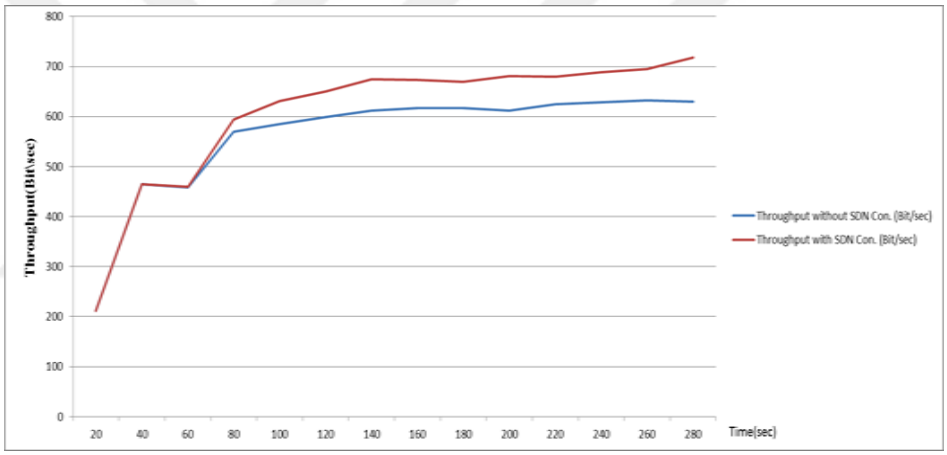

Figure 4.2: Throughput of WSN network with and without SDN controller

4.2.3 Network Received Power

The received power is the power in dBm of the received signal. Figure 4.3 shows the received power consumption of WSN network with controller and without controller. Its clear from figure 4.3 that the received power consumption of WSN network without SDN controller is less than the received power consumption of WSN network. This difference is mainly because two reasons. The first one is one of the basic functions in a SDN architecture is the communication between the control and data plane and an increase in the communication will increase the power consumption the second reason is the power consumption increase when more packets need to be received since the packet reception requires to be turned on longer time than ON time.

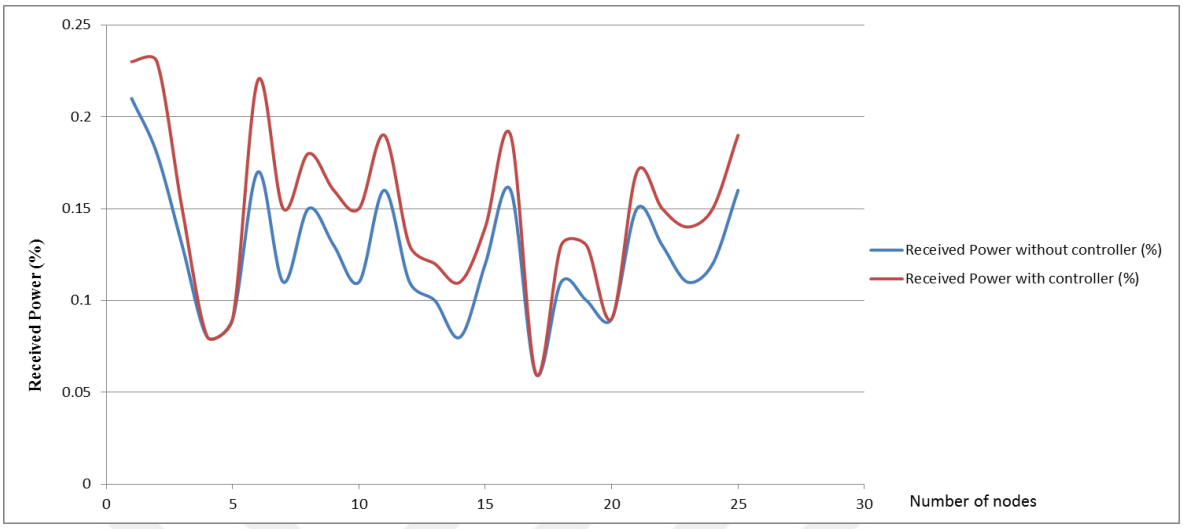

Figure 4.3: Received power of WSN network with and without SDN controller

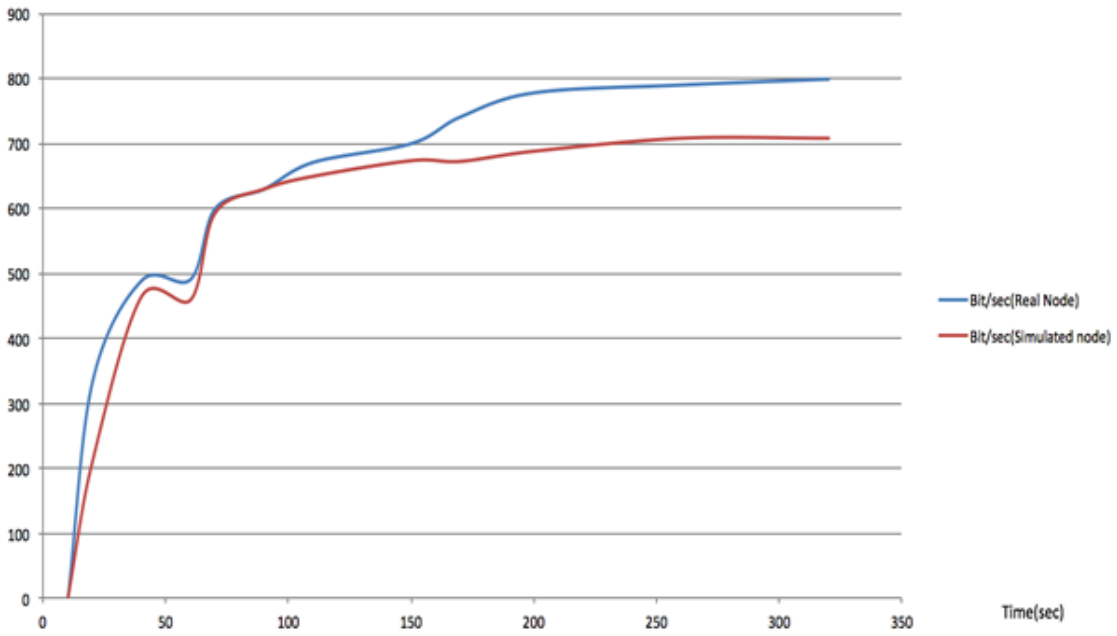

Figure 4.5: Throughput of WSN using Real node and simulated node

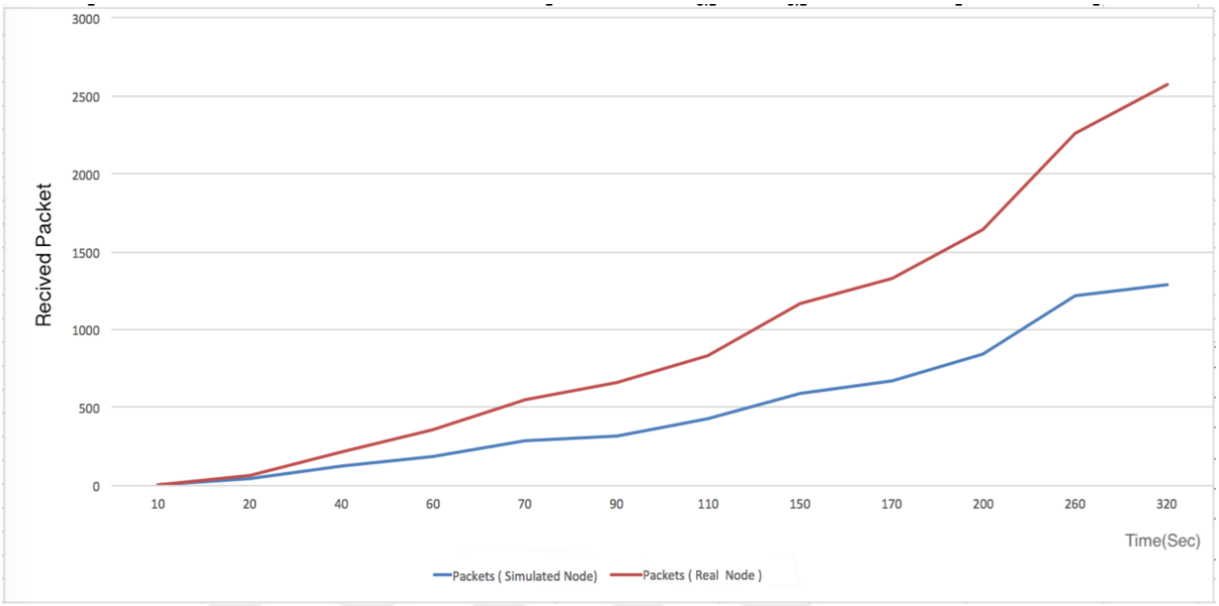

Figure 4.6 : Received packet of WSN using Real node and simulated node

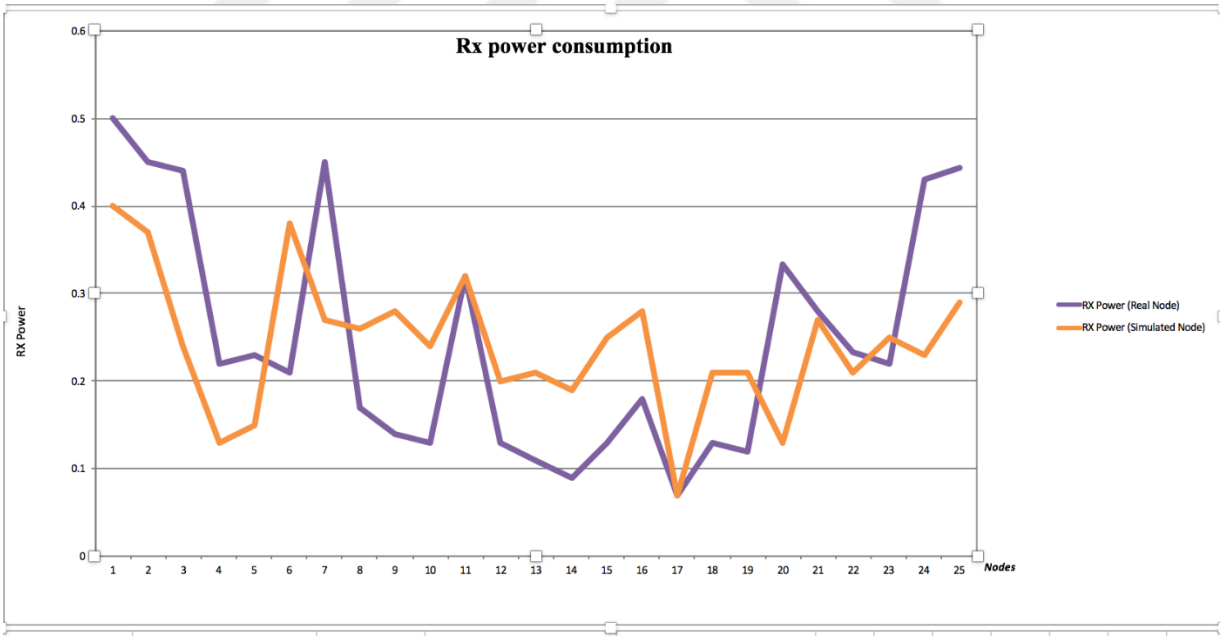

Figure 4.7: Received power of using Real node and simulated node

From figures 4.5 we can see small different in result between Real Network using (Z1 mote) and simulated node in topic throughput . Both part have drawback point is the controller work after 60 sec and the amount of drop packet before 60 similar the drop packet of WSN without

using SDN controller because all node doesn’t know the best path to sink node .But after that the throughput increased in second part when we used Real node

From fig. 4.6 the received packet is increased when we use real node .

From fig 4.7 The received power is not stable .as we see in some part the received power in first part is less than the received power . and in other part received power when we use real node is less that simulation part .

5. Conclusions and Suggestions for Future Works

5.1 Conclusions

In thesis we display how we can implements wireless sensor networks based software define network in this paper we implement the network in two way the first one is by implement network in simulation program and after we implement it we We highlighted some concept :

1. How this approach can simplify management of WSN network.
2. How increase throughput and the received packet of WSN networks.
3. The power consumptions of WSN network increase when we use SDN controller.
4. The receive power is not good all time when we use real node and this very clear in fig. 11
5. Throughput increased when we implement network on real node (Z1 node) this happened due to increased in received packet .

And in the second part we implement the wireless sensor network in real hardware devise (Z1) mote and we see that there is no too much different between two scenarios

5.2 Future Works

1. Demonstrate new method that reduce power consumption in network.
2. We could not find any try effort that takes security into consideration. However, we could identify a few previous efforts on implementing SDN on WSN and on we must add more attention to this part

References

[1] Fei Hu,"Network Innovation through OpenFlow and SDN: Principles and Design", Taylor & Francis Group LLC, Broken Sound Parkway NW, 2014.

[2] Brian Underdahl ,Gary Kinghorn ," *Software Defined Networking for dummies", Cisco Special Edition, John Wiley & Sons, Inc, Hoboken, New Jersey,2015* .

[3] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti,"A survey of software-defined networking: Past, present, and future of programmable networks," Communications Surveys Tutorials, IEEE,vol. 16, no. 3, pp. 1617–1634, Third 2014..

[4] Fahad Kameez , P. Naras himaraja ," Software Defined Networking Architecture and Openflow Network Topologies ", Volume 3, Issue 8, pp: 302-336, August 2015.

[5] T. Benson, A. Akella, and D. Maltz, "Unraveling the complexity of network management," in Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, ser. NSDI'09, Berkeley, CA, USA, pp:335–348, 2009.

[6] I. F. Akyildiz , W. Su , Y. Sankarasubramaniam , Y. , and E. Cayirci, " Wireless sensor networks: A survey ", *Computer Networks (Elsevier) Journal*, Vol. 38 , No. 4 , pp: 393 – 422, Mar. 2002.

[7] I. F. Akyildiz , T. Melodia , and K. R. Chowdhury , " A survey on wireless multimedia sensor networks " , *Computer Networks (Elsevier)* , vol. 51 , no. 4, Mar. 2007 , pp. 921 – 960 .

[8] F. Hu and S. Kumar , " QoS considerations for wireless sensor networks in telemedicine" , in *Proceedings of 2003 Intl. Conf. on Internet Multimedia Management Systems* , Orlando, Florida, pp. 323 – 334 , Sept. 2003.

[9] B. Girod , A. Aaron , S. Rane , and D. Rebollo - Monedero , " Distributed video coding " , *Proceedings of the IEEE* , vol. 93 , no. 1, pp. 71 – 83, Jan. 2005.

[10] Jun Zheng,Abbas Jamalipour," WIRELESS SENSOR NETWORKS A Networking Perspective", John Wiley & Sons, Inc, New Jersey,2009.

[11] Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal," Wireless sensor network survey", 7 April 2008

[12] I. Howitt, J.A. Gutierrez, IEEE 802.15.4 low rate-wireless personal area network coexistence issues, Wireless Communications and Networking 3 (2003) 1481–1486.

[13] ZigBee Standards Overview,<http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=01J4Fs25657725>,Nov 10 2010.

[14] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, "Transmission of IPv6 packets over IEEE 802.15.4 networks", RFC 4944, Internet Engineering Task Force RFC 4944, September 2007.

[15] Salvatore Costanzo, Laura Galluccio, Giacomo Morabito, Sergio Palazzo," Software Defined Wireless Networks (SDWNs): Unbridling SDNs",2012.

[16] Tie Luo, Hwee-Pink Tan, and Tony Q. S. Quek,"Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks", Institute of Electrical and Electronics Engineers (IEEE), vol.16,no. 11,pp. 1896- 1899, 2012

[17] Martin Jacobsson , Charalampos Orfanidis," Using Software-defined Networking Principles for Wireless Sensor Networks", *May 28-29, 2015*.

[18] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable

networks," Communications Surveys Tutorials, IEEE, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[19] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in the Fourth International Symposium on Information Processing in Sensor Networks (IPSN'05), Los Angeles (CA), USA, Apr. 15, 2005.

[20] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Mar. 2008.

[21] M. Jacobsson, C. Guo, and I. G. Niemegeers, "A flooding protocol for manets with self-pruning and priorited retransmissions," in the International Workshop on Localized Communication and Topology Protocols for Ad hoc Networks (LOCAN'05), Washington DC, USA, Nov. 7-10, Nov. 7-10, 2005.

[22] A. Elsts, F. H. Bijarbooneh, M. Jacobsson, and K. Sagonas, "Demo abstract: ProFuN TG: A tool using abstract task graphs to facilitate the development, deployment and maintenance of wireless sensor network applications," in the 12th European Conference on Wireless Sensor Networks (EWSN'15), Porto, Portugal, Feb. 9-11, 2015.

[23] F. Aslam, C. Schindelhauer, G. Ernst, D. Spyra, J. Meyer, and M. Zalloom, "Introducing TakaTuka: a java virtualmachine for motes," in the 6th ACM conference on Embedded network sensor systems (SenSys'08), Raleigh (NC), USA, Nov. 5-7, 2008.

[24] P. D. Mil, B. Jooris, L. Tytgat, J. Hoebeke, I. Moerman, and P. Demeester, "snapMac: A generic MAC/PHY architecture enabling flexible MAC design," Ad Hoc Networking, vol. 17, pp. 37–59, Jun. 2014.

[25] G. Morabito. “SDN-WISE: A SDN solution for Wireless Sensor Networks”. Proc. of INFOCOM 2015. April 2015.

[26] The Contiki Project, "Contiki: The Open Source Operating System For the Internet Of Things," < http://www.contiki-os.org/ >. July 2012.

[27] A. Dunkels, B. Gronvall and T. Voigt, "Contiki -a Lightweight and Flexible Operating System for Tiny Networked Sensors," in Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I), Tampa, Florida, USA, 2004.

[28] Christopher Pinola,”Evaluating the Performance of Synchronous and Asynchronous Media Access Control Protocols in the Contiki Operating System”, worcester polytechnic institute, Computer Science,2012.

[29] Abhinav singh,” Instant Wireshark Starter”, Packt Publishing Ltd, Birmingham B3 2PB, UK, January 2013.

[30] Brad Hedlund ,“How to Calculate TCP throughput for long distance WAN links”, < http://www.bradhedlund.com/>. Dec 19, 2008 .