T.C.
SAKARYA UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY

INTEGRATING SOFTWARE DEFINED NETWORKING APPROACH INTO WIRELESS MULTIMEDIA SENSOR NETWORKS

Ph.D. THESIS

Ali Burhan ALSHAIKHLI

Department	**:**	**COMPUTER AND INFORMATION ENGINEERING**
Supervisor	**:**	**Prof. Dr. Celal ÇEKEN**

November 2018

SAKARYA UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY

INTEGRATING SOFTWARE-DEFINED NETWORKING APPROACH INTO WIRELESS MULTIMEDIA SENSOR NETWORKS

Ph.D. THESIS

Ali Burhan ALSHAIKHLI

Department	:	**COMPUTER AND INFORMATION ENGINEERING**
Supervisor	:	**Prof. Dr. Celal ÇEKEN**

This thesis has been accepted unanimously by the examination committee on 28.11.2018

[illegible]	[illegible]	[illegible]
Prof. Dr. Celal Çeken	**Assoc. Prof. Dr. Kerem Küçük**	**Assoc. Prof. Dr. Emine Doğru Bolat**
Head of Jury	**Jury Member**	**Jury Member**

[illegible]	[illegible]
Assist. Prof. Dr. Seçkin Arı	**Assist. Prof. Dr. V. Harun Şahin**
Jury Member	**Jury Member**

DECLERATION

I declare that all the data in this thesis was obtained by myself in academic rules, all visual and written information and results were presented in accordance with academic and ethical rules, there is no distortion in the presented data, in case of utilizing other people's works they were refereed properly to scientific norms, the data presented in this thesis has not been used in any other thesis in this university or in any other university.

Ali Burhan Al-Shaikhli

21.11.2018

ACKNOWLEDGMENT

I would like to show gratitude and praise to Allah for his blessings and favors. Secondly my thanks and appreciation go to my advisor Prof. Dr. Celal Çeken for his unprecedented support and priceless advices from the first day we met. And I would like to acknowledge Mohammed Hussein for his work with us in the project. All people in the department and the institute were very helpful, I would like to thank them for that.

Also, I would like to extend my gratitude to my father (may Allah grant mercy upon him) for his support over the years, my mother for her kindness and prayers, my wife for giving me a reason to keep moving forward, all friends and colleagues who backed me up throughout my journey.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

SDN	: Software Defined Networking
WSN	: Wireless Sensor Networks
WMSN	: Wireless Multimedia Sensor Networks
SDNC	: SDN Controller
ED	: End Device
WSAN	: Wireless Sensor and Actuator Networks
MAC	: Medium Access Control
TIMAC	: Texas Instruments MAC
ISR	: Interrupt Service Provider
ARM	: Advanced RISC Machine
API	: Application Programming Interface
TI	: Texas Instruments
GoP	: Group of Pictures
FFMPEG	: Fast Forward Motion Picture Experts Group
PSNR	: Peak Signal to Noise Ratio
SSIM	: Structural Similarity
CODEC	: Coding Decoding
LR-WPAN	: Low Rate Wireless Personal Area Network
SO	: Superframe Order
BO	: Beacon Order
TDM	: Topology Discovery Mechanism
CoG	: Center of Gravity
MCU	: Micro-Controller Unit
TIMAC	: Texas Instrument Medium Access Control
ZAP	: Zigbee Application Processor

ZNP	: Zigbee Network Processor
OSAL	: Operating System Abstraction Layer
HAL	: Hardware Abstraction Layer
UART	: Universal Asynchronous Reciever/Transmitter
EED	: End to End Delay
EVM	: Evaluation Module

LIST OF FIGURES

LIST OF TABLES

SUMMARY

Keywords: WMSN, SDN, SDN Controller, Wireless Sensor Networks, Software-Defined Networking, Embedded Systems.

This research study addresses some critical issues of wireless sensor networks and comes up with a proposition for tackling these issues by means of a trending notion in networks that is SDN. SDN standardization exertions are in progress through active IEEE standards projects, since SDN has revealed new perspectives in managing existing networks and apply its practicality and interoperability.

In this thesis, SDN is presented in a way it would be a highly suitable choice for WSAN type of networks, concentrating on abstracting the network and simplifying management for the sake of less overall energy consumption and application-specific adaptation. The proposed architecture has been modeled and simulated using Riverbed Modeler Software for performance evaluation. Additionally, the simulated architecture was developed for experimental devices implementation. TI network processor of ARM-based CC2538EVM module with an application processor of smartRF06EB board was used in the development. The network processor is used either as a stand-alone only-forwarding node or mounted on the application processor to utilize existing sensor data, buttons and other interfaces in smartRF06EB.

In the developed architecture simulation, several demonstrations were carried out using various types of traffic. Furthermore, the model included a centralized topology-aware routing mechanism embedded in SDNC. At the same time, a traditional protocol framework simulation was also conducted for comparison purposes. Results presented a very good enhancement in power consumption aspect of view and slightly better results in terms of delay and throughput.

KABLOSUZ ÇOKLU ORTAM ALGILAYICI AĞLARINA YAZILIM TANIMLI AĞININ YAKLAŞIMI

ÖZET

Anahtar kelimeler: WMSN, SDN, SDN Denetleyici, Kablosuz Algılayıcı Ağlar, Yazılım Tanımlı Ağlar, Gömülü Sistemler.

Bu araştırma çalışması, kablosuz algılayıcı ağların bazı kritik konularını ele almakta ve bu konuların çözümü için, bilgisayar ağlarında yeni bir kavram olarak ortaya çıkan yazılım tanımlı ağ yaklaşımını önermektedir. Yazılım tanımlı ağ yaklaşımının mevcut ağları yönetmede yeni bakış açısı ortaya koyması, pratiklik ve birlikte çalışabilirlik özellikleri nedenleriyle, aktif IEEE standartları projelerine paralel olarak standardizasyon çalışmaları devam etmektedir.

Bu tez çalışmasında, yazılım tanımlı ağ yapısının, daha az enerji tüketimi ve uygulamaya özel adaptasyon için ağın soyutlanması ve yönetiminin basitleştirilmesi özellikleri üzerine yoğunlaşılarak, kablosuz algılayıcı/eyleyici ağlara uygun olduğu gösterilmiştir. Başarım değerlendirmesi için, önerilen mimarinin modeli ve benzetimi Riverbed Modeler Yazılımı ile gerçekleştirilmiştir. Ayrıca, oluşturulan modelin fiziksel olarak gerçeklemesi de yapılmıştır. Geliştirme çalışmasında ağ işlemcisi olan ARM tabanlı CC2538EVM modülü smartRF06EB uygulama işlemcisiyle birlikte kullanılmıştır. Oluşturulan topoloji içerisinde ağ işlemcisi hem tek başına hem de, mevcut algılayıcılarının, butonlarının ve diğer arayüzlerinin kullanılabilmesi için smartRF06EB borduyla birlikte kullanılmıştır.

Gelişmiş mimarinin benzetimi için çeşitli trafik türleri içeren senaryolar gerçekleştirilmiştir. Oluşturulan mimari içerisinde yazılım tanımlı ağ denetleyicisine gömülü merkezi bir topoloji-farkında yönlendirme mekanizması yer almaktadır. Aynı zamanda, karşılaştırma amacıyla geleneksel bir protokolün benzetimi de gerçekleştirilmiştir. Benzetim sonucunda, güç tüketimi açısından çok daha iyi, gecikme ve verimlilik açısından biraz daha iyi gelişmeler elde edilmiştir.

CHAPTER 1. INTRODUCTION

This chapter presents an introduction to topics that are covered in the thesis. A problem statement will specify the main issue this research is trying to tackle. The chapter continues to state the aim of this study and the contributions to achieve research goals. Furthermore, a number of related studies are presented then the organization of this thesis is outlined at the end of this chapter.

1.1. Problem Statement

Wireless sensor networks (WSNs) struggle due to energy constraints since each device in the network has limited battery capacity and, most of the time, a device battery is unlikely to be replaced after being deployed[1][2][3].

The research work of this thesis is intended to investigate if wireless sensor networks in some specific fields of applications would be better performing in case of integrating a trending concept of Software-defined Networking. Traditional sensors have been autonomous in terms of network decisions such as routing [4], resulting in a lot of overhead traffic, unmeasured energy consumption and inaccurate decision leading to unwanted delays. There has been plenty of research aiming at centralization architectures for sensor networks to overcome these problems, this study targets a promising trend of SDN and proposes a new architecture for sensor networks to enhance its performance.

Applications and uses of wireless sensor networks require taking into account a specific characteristic of the designated network. A number of these characteristics are discussed here [5].

As the number of deployed sensors grows, sensors get more constraints in resources. This makes sensors suffer from the lack of being loaded with more than one application. This type of network can be called a stiff network. The approach presented in this research work proposes a protocol of reprogramming sensor devices with a new application according to network demands by means of flow tables. The result of that is a demoted end device with minimal resources to act as forwarding nodes working at the data plane as SDN paradigm states.

Another aspect of a large number of deployed sensors is the large amount of traffic that can be produced and the accuracy in network decisions i.e. routing. A way to overcome such an issue is by a protocol that demands a bird's eye view of the whole network for a greater ability to get most of the whole network resources. This would be followed by overall power conservation and network life-prolonging in addition to precision in network decision.

1.2. The Aim Of The Research

The research that conducts this thesis aims at achieving simplified network configuration and management and accordingly, lower cost network. Also presenting an adaptation to network/application change and application-based QoS support.

WSAN development tends to provide flexible enhanced network structures where there will be no need of machine-human contact, instead of that, a network structure where devices and machines communicate, collect data and create a knowledge base to make decisions regarding network management.

1.3. Research Contributions

To accomplish the aim of our research, a new SDN controller protocol stack is developed which contains network intelligence with the capability of establishing the network and communicating with other sensor nodes. Beside that, a messaging protocol is developed. And for creating and managing network intelligence, algorithms are added to the protocol. The most important part of network intelligence is the decision support system module. The algorithms in this unit will be used for the resolution of multi-parameter decision-making problems.

The variables created by the proposed architecture will be modeled using a realistic performance evaluation modeler (OPNET) for the performance evaluation. And for the implementation of the simulation model, Libelium Waspmote and TI cc2538 Texas instruments devices will be used to develop an experimental testbed.

1.4. Motivation

The rapid development in microelectronics and wireless communication technologies has led to the production of small size, low cost, low power consumption, mobile and multifunctional sensor nodes constructing Wireless Sensor Networks (WSN). At first, WSN used only for environment monitoring but with the new actuator nodes, system inspection took an even wider applications field that led to Wireless Sensor and Actuator Networks (WSAN). Industrial, medical, military and environmental monitoring, observing, collected data processing and decision-making functions resulted from the use of WSAN has brought numerous applications to those fields.

One of the major problems in traditional WSAN structures is that they require application specific and complex network design and management operations.

Despite the fact that WSNs are well known for their effortlessness of deployment and low cost, it has been a tough task to manage them due to resource underutilization characteristic. Nevertheless, with IoT envisioned to be a worldwide phenomenon, sensor devices are anticipated to be deployed on a large scale. That requires new emerging protocols to govern a network with such a size [6].

In the last few years, a new favorable trend, Software Defined Networking (SDN) has been raised by computer networks communities as a new solution for abstracting physical network structure from applications.

The SDN concept strategy to achieve extensible and easy to manage computer networks is considered a strong motivation to apply the same principle to sensor networks. Besides, SDN also is about utilizing the centralization concept in network design which is proved to be a good way to improve sensor networks. Furthermore, introducing some level of programmability as tools to give more authority to system administrators and network engineers to better manage and optimize networks [7].

1.5. Related Work

Authors in [8] leverage SDN-WISE testbed to integrate switches networks with sensors networks to build an IoT environment. The testbed was implemented for TI CC2538 devices in addition to Mininet emulation modules. Results show that the integrated approach results in less communication among the nodes.

A master thesis [4] proposed an SDN based protocol. It was compared with a traditional routing protocol, Control Tree Protocol (CTP) that was implemented in TinyOS. The thesis shows that SDN presents adaptability to variable conditions of node operations in contrast to static reprogramming nodes to change behavior.

A journal article [9] uses SDN, network functioning virtualization NFV, and cloud computing together with a 6LoWPAN gateway to add abstraction and liability to network administration. This work took advantage of Arduino pico internet protocol stack for 6LoWPAN protocol and MATLAB software for end-user app. Results showed improvement in network lifetime and network discovery delay in contrast to conventional 6LoWPAN nodes.

This article [10] presents an improved SD-WSN framework to merge WSN into industrial IoT. Mininet emulator and Floodlight controller was used in this work. Results revealed a good improvement over traditional WSN but a slight improvement over traditional SDN in terms of energy and time delay.

This paper [11] uses SDN to reconfigure sensor nodes to identify probable attacks in Mobile WSNs such as selective forwarding, with the aid of fog computing to recognize intrusion patterns. CloudExp simulator was used to evaluate the presented model. The approach showed a feasibility in stopping malicious sensor nodes with reasonable network load overhead.

In [12], an improvement for an SDN-WISE framework is introduced using Fuzzy Topology Discovery Protocol to enhance network performance in the aspects of packet arrival, packet loss, and energy. A fuzzy protocol is used in the decision of selecting the finest forwarding node for each node considering a number of neighbors, queue length, and energy.

The thesis in [13] provides a power consumption model for WSN time slotted channel hopping networks specifically for devices running OpenWSN which is a modular ecosystem designed for IoT. The model considers network-related CPU state changes to predict energy consumption, making this model suitable for simulations and experimenting. The model can be accumulated to OpenSim simulator in OpenWSN.

Authors in [14] proposed an energy-efficient algorithm depending on multi-energy-space for software-defined wireless sensor networks. The proposed platform is simulated in MATLAB and compared with other traditional energy-efficient routing algorithms such as Wireless Routing Protocol WRP, Energy-aware Temporarily Ordered Routing E-TORA and Low Energy Adaptive Clustering Hierarchy Centralized LEACH-C.

This paper [15] is one of the early attempts to explore the beneficialness of SDN paradigm to networks like sensor networks that has particular design specifications. In addition to investigating the possibilities of using SDN with IEEE 802.15.4 standard protocols. It introduces a generic structure of SDWN with some design details and parameters.

The paper in [16] proposes an IoT multi-network controller layout to deal with network impairments resulted from heterogeneity and diverse applications. The research work implemented a prototype using Qualnet simulation framework. An OpenFlow-like protocol was inserted into the IP layer. For each scenario, one node acts as a controller, all other nodes as controlled.

Another early exploratory proposition research study is brought up by [1]. Authors put an argument for an untested base station structure for sensor networks based on software-defined networking idea. The research includes explanations about structural parts of the proposed layout with some discussions to tackle challenges and opportunities, yet no demonstration of any type is presented.

Authors in [17] suggest a platform model as a complete software-defined solution for IoT network called SDIoT. The solution consists of SDN, software-defined storage SDStore, and Software-defined security SDSec. The model is meant to overcome challenges in IoT structure to efficiently send, store, protect exchanged data in the IoT network. The paper didn't present an experimental layout of any kind.

A discussion has been held by [18] to look into standardization synergy between SDN based WSN and IoT. This paper examines RPL [19] and before-mentioned TinySDN protocols, investigate the feasibility of them inter-operating to facilitate the design and establishment of IoT structures. Also, discusses particular issues like network management, routing, and resource sharing.

[20] presents a multiple controllers structure of TinyOS SDN architecture. It takes advantage of SDN based sensor node types which is end devices and switches. The platform was tested in COOJA simulator and TelosB devices on TI MSP430 emulator. Results are shown for the delay and memory usage.

The paper in [21] suggests a measurement layout for WSN called TinySDM which is software defined. The research resulted in designing a C-like language to customize measurement jobs. The study discussed: packet pathfinding, measuring delay and metric gathering while demonstrating TinySDM architecture.

Authors in [3] came up with a routing algorithm that is claimed to be energy-efficient for SDWSN. The network consists of a main control server, control nodes (cluster heads) and other sensor nodes. The core contribution is cluster head selecting using PSO algorithm to deal with it as an NP-hard problem. Simulations are done using MATLAB compared with a conventional LEACH routing algorithm.

A sleep scheduling algorithm which is SDN-based is presented in [2], to extend the lifetime of a network. Simulation results are compared with a classical EC-CKN algorithm results. For a static environment, the proposed algorithms outperform the transitional one, yet no dynamic conditions for WSN are tested.

Authors in [22] proposed a solution for WSNs to embed a MapReduce paradigm which is a mass data processing framework by means of SDN concept. The basic work of the research depends on a recent SDN-WISE platform to use its functionalities which permit the dynamic loading and execution of user-defined MapReduce tasks in sensor nodes.

A hardware-in-the-loop was utilized in [23] to transfer data from OPNET simulation model to the internet using a SITL model. The system applies SDN concept to emulation framework. The authors made the controller manages the hardware layer by southbound interface of the cloud data center to control its router supporting SDN.

Authors in [24] put forward a routing protocol that is based on SDN notion for WSN. The research concentrated on several aspects of conducting results: delay, power usage, memory, and security. Results are compared with those of CTP and TinySDN protocols.

A study [25] compares SDN based solution for WSN with other two solutions by terms of delay and packet loss ratio in a static, quasi-static and dynamic conditions of WSN.

The study used complete hardware testbeds for all three solutions. The experiments conducted using 53 nodes in the experimental platform of flextop.

Authors in [26] proposed an SDN-based protocol that is supposed to be traffic aware, which means it suggests a solution for congestion control in WSNs. It takes advantage of the SDN flow tables to dynamically reprogram the nodes on the fly. The framework performance is evaluated in contrast to traditional network protocols.

Authors in [27] propose a load balancer for M2M networks with traffic identification by the use of an SDN network (Controller and OpenFlow 1.0 switches) and sensor network (relays and sensors). The aim is to collect sensor data from a sensor network then transfer it to a normal network through an M2M gateway and tackle the load balancing issues. The proposed solution focused on one-way traffic.

1.6. Thesis Outline

This thesis consists of six chapters, the first one introduces the topic this thesis is dealing with in the aspects of the problem statement, the aim of research and contributions with some literature survey. The second chapter goes beyond concepts of SDN to see how SDN is good for sensor networks. The third chapter discusses a case of video application through traditional sensor networks with some basics about video traffics.

The fourth chapter presents the proposed SDN-based platform details, simulation environment, results, and discussion with a conclusion regarding the simulation. The fifth chapter talks about the implementation of the proposed platform in TI devices and how the test bed was prepared then shows some results and discussion then concludes accordingly. The last chapter is a small chapter with an overall conclusion of the research word and expected future trends regarding the main topic.

CHAPTER 2. SOFTWARE DEFINED NETWORKING: CONCEPTUALIZATION AND IMPACT

This chapter gives a clear explanation about SDN concept and underlying topics regarding architecture, protocol stack, and planes. It also relates to how SDN can be a suitable solution for sensor networks.

2.1. SDN Essentials

Classical IP networks normally consist of various network elements like routers, switches and other devices that are specified by an application. Some vendor specific instructions should be followed to manage and configure the network in this situation, which imposes a setup cost plus maintenance. Therefore, managing huge size network of such type would be a difficult task and susceptible to many errors.

Traditional networks, in some applications, can be perceived as suffering from crucial drawbacks such as consistency, resiliency, scalability, and controllability. The growth of the internet, its applications, social networking, cloud services, and virtualization, demand and require networks with efficient bandwidth usage, efficient accessibility and above all reliable in dynamic environments. The development towards these network requirements has become an essential issue to adapt to the new application-specific environments [28][29].

SDN is considered one of the rising trends in state-of-the-art networks as a solution for the recent related issues and challenges faced by these networks. However, at the time of these writings, researches still in progress for the most effective employment of this concept, mostly because of its unconventionality in networks world. We can define a network as a platform consists of interacted, attached or associated machines and codes provided and handled by a person or a device. Thus, it is implied by the new advanced networks to actualize a reliable network by means of computing paradigm just as SDN [30].

2.1.1. SDN architecture

The basic idea of SDN is separating the control plane from the data plane and make network managers dynamically administer a huge number of devices and applications. To accomplish that, network administrators use APIs to handle services, traffic needs, Quality of Service and packet forwarding policies. These APIs are written in high-level programming languages and are suitable for multi-vendor heterogeneous devices. Thus, SDN basic way of working can be depicted as three main components or layers: data plane, control plane, and applications as it can be seen in Figure 2.1.

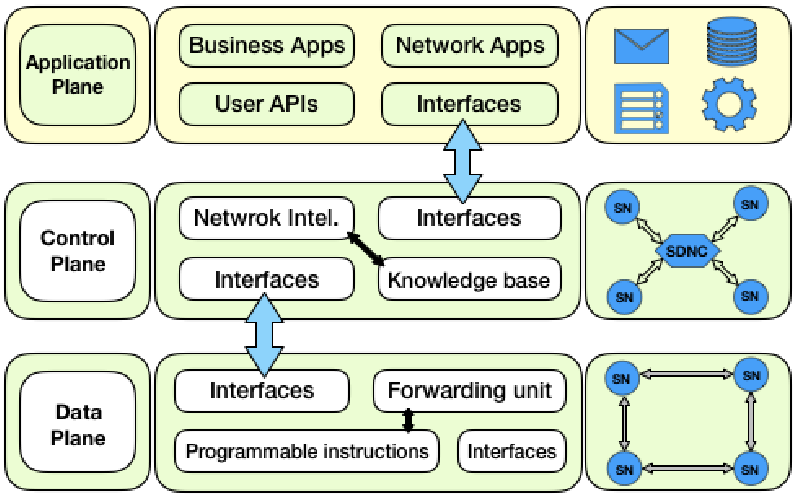


Figure 2.1. SDN architecture

According to the application needs, a specific API is determined and added to the system which will specify a set of instructions. The control plane translates these instructions into rules that are then transferred to the entities in the data plane. The entities in the data plane are the devices that deal with data streams directly, and the control plane is the SDN controllers that have services in accordance with the application needs [31].

The control plane and the forwarding plane are continuously communicating. Open Networking Foundation (ONF) has been maintaining a set of specifications regarding OpenFlow as an instance of SDN architecture. OpenFlow is how the control plane represented by the SDN controller and the data plane represented by forwarding devices are interacting with each other.

Although this would seem to enhance the networking procedures and operations by converting complicated networking tasks from forwarders to controllers, there are some anticipated challenges and issues that are expected or seen probable in such platforms such as: available frameworks stable transition to SDN platform, suitable hardware for broad deployment and other issues that will be discussed in some detail later on this chapter [32].

2.1.2. OpenFlow protocol stack

OpenFlow has been standardized by ONF [33] for the global network (internet) linking controllers and forwarding devices. It comprises some of the often-employed forwarding plane protocols of datalink, network and transport layers [34].

OpenFlow is a protocol in which it permits making changes to the forwarding plane by SDN controller. As OpenFlow is considered flow based, forwarding devices in the data plane retain a flow table that includes flow entries, and those decide how to deal with incoming packets. a flow entry includes match rule field, actions, and statistics. Figure 2.2. (OpenFlow architecture) illustrates the protocol and its basic functionalities.

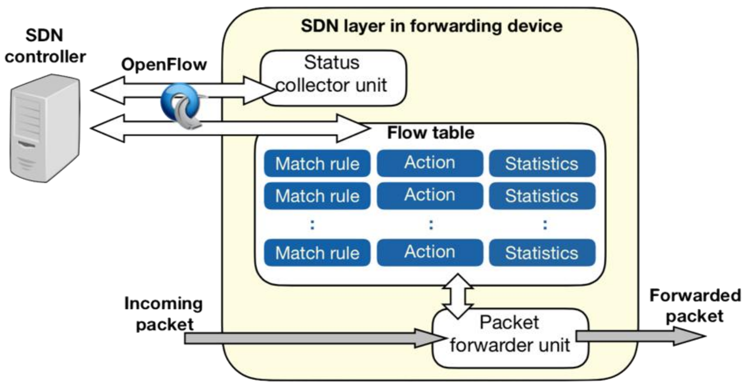


Figure 2.2. OpenFlow architecture

The match rule compares certain values determined by the SDN controller with values obtained from incoming packets. According to what results from the match rule, actions will be taken and executed consequently. For each incident, statistics will be calculated and checked correspondingly for other actions to be made. In case of no flow entry found for an incoming packet, action will be dropping the packet or send it to SDN controller to set new rules for new flow entries [35]. Figure 2.3. shows a flowchart of how SDN based forwarding devices deal with incoming packets.

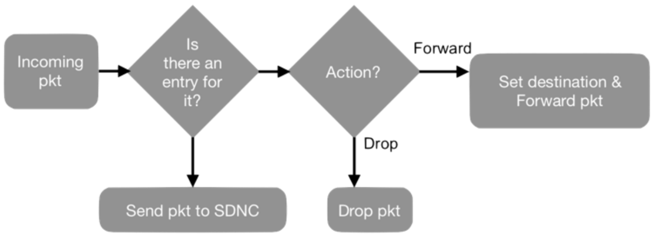


Figure 2.3. Packet flow in SDN enabled forwarding device

Three types of messaging are specified by Openflow between controller and forwarding devices, and they are: (1) controller to forwarding device, which might me requests from controller, orders or other instructions to forwarding devices, (2) Asynchronous, that are messages sent by devices to controller without inquiring, it can be rule-requests, reports or errors, (3) symmetric messages in either way like handshaking during network establishment [36].

2.1.3. Planes of SDN paradigm

As mentioned before, SDN essence is the dissociation between the control plane and forwarding plane for the sake of building hardware unrelated to lower forwarding entities. In classical networks, devices in the forwarding plane is in charge of receiving packets, checking routes for them, altering packet headers then forwarding them to the required port. whilst in SDN, this entity’s layer will be adjustable, therefore, it would be able to do much more complex functions with traffic like controlling access, packet header mapping, watching traffic, traffic-aware routing, events arrangement, deep inspecting packets, and smart packet forwarding.

In addition to that, the control plane is created to have the intelligence of the network. SDN controller at the control plane collects knowledge, details, and particulars concerning devices in the network and how they are interconnected. In accordance with that, it constructs flow tables as an output, of a decision derived from bird’s eye view of the SDN controller over the whole network [37].

2.1.3.1. Data plane

Data plane located at the bottom of the SDN architecture illustration. It consists of network components and what is called SDN Datapaths [38] which are logical entities that interact with the control plane using Control-Data-Plane Interface (CDPI). The management and admin plane have access to all planes of the SDN architecture (application, control, and data planes) is in charge of installing network components and mapping SDN Datapaths.

In sensor networks world, the basic entities of the forwarding plane’s infrastructure are sensor devices. A sensor device is a combination of physical parts and programs. Fundamentally, it includes a power source, a sensing module, and wireless module. Programs are mainly codes reading sensing values, network processing, and in SDN, SDN layer, which consists of instructions from SDN controller and codes for interacting with SDN controller [35].

The principal task of the data plane is forwarding incoming packets. The modifiability of data plane adds a plenty of potentials to the platform like deep packet inspection, transcoding, detecting abnormalities, and traffic-oriented decision support algorithms [29].

2.1.3.2. Control plane

The control plane is logically not necessarily physically centralized, and it’s detached from the forwarding plane. it outlines an overall system status for administration and converts application requests to data-level instructions. control plane could be dispersed over several actual controllers in a way that they collaborate to accomplish applications’ tasks. Control plane makes decisions based on an updated comprehensive state of the network instead of a confined response of individual node [38].

SDN controller communication through NorthBound Interface with the application layer, and through SouthBound interface with the data plane layer. Hence, it transforms application requests to lower-level instruction, and inversely, provide upper layers with pertinent information to keep application plane updated about the network [38].

From sensor networks point of view, the control plane contains a network intelligence module, network topology mapper module, flow table module, and interfaces. The basic tasks are: keeping an updated topology map, flow table creation, and managing interfaces. For obtaining network state, it uses reporting messages to get sensor device battery state, distances, neighbor list, link status, and other values. Mapper module uses this information to produce a network topology map (bird’s eye view). Therefore, the decision that is made based on such information would be as accurate as it can be [35].

2.1.3.3. Application plane

SDN application layer is a collection of programs that require some network tasks from the framework in a programmatic way to get the demanded network response. For that purposes, it uses NorthBound Interfaces NBIs to send its demands to control plane which in return will translate application layer requirements into lower-level instructions to shape forwarding plane entities in the interest of achieving applications plane demands. So, SDN application includes SDN application logic and a number of NorthBound Interface drivers [38].

2.2. SDN versus classical network (SDN-less network)

The above-explained SDN based network paradigm is compared to some traditional network paradigms in terms of working specifications, realization, administration, control, originality, fault tolerance, maintenance, and setup.

SDN separation of control and forwarding planes made it effortlessly managed and employed, while classical one's coupling of data and control functions made them rigid and complexly managed.

The easy and quick deployment of SDN network devices is recognized over the classical networks which needed a lot of effort for deployment. Also, easily managed and modified via APIs whereas traditional networks need to be managed individually.

New applications can be simply installed in case of new network demands in SDN based platform, whilst a simple change in non-SDN networks may require a new structural design.

As the control plane keeps an updated overall map of the network, network faults can be detected quickly and be avoided or overcome efficiently. Faults need to be addressed manually in SDN-less networks [39].

In spite of that, the decision of SDN deployment in the existing networks is hard to take. Many issues still open for testing and research concerning interoperability of multi-vender's devices and security.

2.3. SDN For Wireless And Mobile Of Networks

SDN was introduced at first as a wired networks solution. The concept confirmed to be effective in wired networks, what made researches look into applying it to other types of networks such as but not limited to wireless, mobile, sensor networks. The resulted outcomes from applying SDN was network coverage improvement, connectivity, implementation cost-effective network upgrade [40].

One of the first pioneering projects to apply SDN to WLANs was OpenRoads [41] from the University of Cambridge. SDN based solution was deployed in Cambridge university campus network, focused on OpenFlow open source framework. For mobile networks, CellSDN [42] claim that SDN can make cellular networks implementation and control. At the same time, it addresses scalability challenges for SDN in cellular networks structures.

2.4. SDN Vulnerability

SDN security is an important issue to be addressed in SDN network design. There has not been a security risk management that specify SDN security issues and the effects of its particular features. Salaheddine Zerkane et al [43] made vulnerability analysis in order to address these deficiencies and estimate their influence.

A number of standardized and conditional issues that need to be dealt with in the design of SDN based networks, such as infrastructure cost to support SDN, skills required by network administrators, frameworks and software required and security issues related to the network.

In addition to the risk transforming of the existing network structures from the previous platform to the new SDN based one and initiating new protocols and action plans and applications to harness the best of SDN. Thus, it is vital to comprehend every task module involved in each part required to prepare a new technology deployment [44].

2.5. SDN Debate And Viability For WSNs

Targeting certain features that are associated with WSNs such as resource restriction, process management, complicated patterns, distinct topologies, application-oriented platforms, and routing priorities, SDN works toward getting appropriate control algorithms to sensor networks. The SDN solution suggests simplified programmable network management without the need for reconfiguring sensor devices. The most important strategy to apply SDN solution is keeping an overall network structure image at an intelligent centralized unit (e.g. SDN controller). That would add important potentials to the network like easy network understanding, easy specific network problem solving, easy providing future demands, easy resources access, and improved computing procedures [31].

CHAPTER 3. VIDEO OVER WIRELESS SENSOR NETWORKS

This chapter presents a case study of a well-known protocol (ZigBee) based WSN network with an application of multimedia stream type. Therefore, it starts with some basics about multimedia streams and video, how to represent video data for network evaluation. Then illustrates the case study with the topology and a sequence diagram of the workflow. Results and discussion are presented and the conclusion is drawn at the end of the chapter.

3.1. Video Traffic Representation For Network Evaluation

A video is a type of a widely used piece of information, it has some certain features that made it hard to deal with when transmitted through networks. Its core contents are a sequence of images like at least 25 images sequence in one second. Images are converted to transmittable form by getting a number (or set of numbers) for each dot in the image. The more dots we define in an image, the clearer the image would be. So, a certain bandwidth is needed to transmit these numbers to get an image transferred through a specific medium during a demanded time. For video, multiply that bandwidth by a number of frames per second to get a video of the same image properties during the same demanded time. The outcome is a huge amount of required bandwidth which is most of the time expensive. Video compression techniques mitigated/attenuated the situation by making the same video to be carried using much less bandwidth without quality loss or with a reasonable quality loss [45].

There has been a lot of video compression concepts, the most known one is creating three types of video frames from the original frames denoted as: I, P and B. I frame is created by intra encoding the original frame without any reliance on other frames. P is constructed by encoding the differences of the original corresponding frame and the previous one, in addition to the intra encoding stage. The B frame is constructed the same as P frame but the differences are encoded for the preceding and following frames. The encoding is done in a way so that it can be decoded again on the other side of the transmission. Therefore, there has to be at least one I frame to construct P frames and one I or P frame to construct B frame [46]. Figure 3.1. illustrate how a video frame sequence may be arranged in what is called Group of Pictures GoP, which is a pattern of how frame types: I, P and B are arranged and how often it is repeated. The GoP number is not related to the number of frames per second attribute of the video. Some of the Famous video encoding standards are H.264/ MPEG-4 Part 10, H.262/MPEG-2, MPEG-1 and H.261 [47].

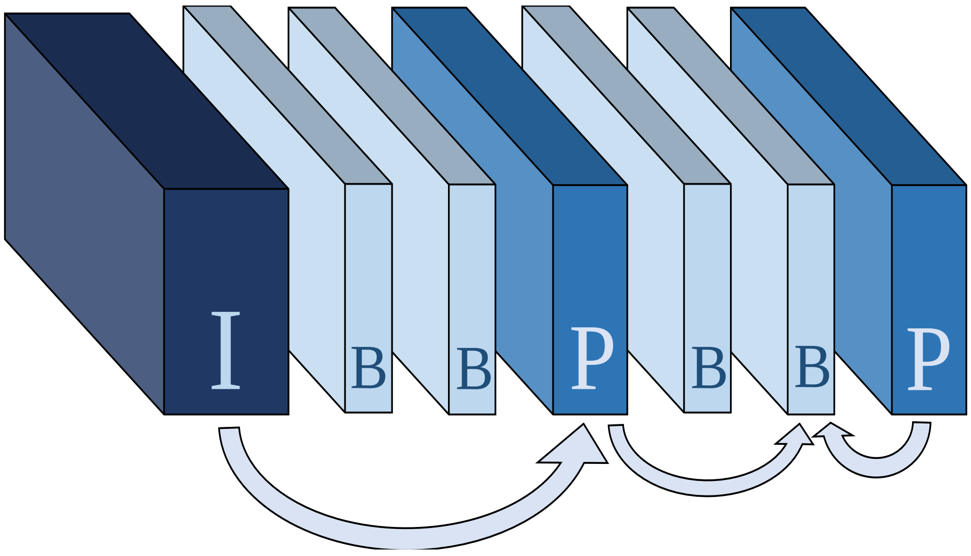


Figure 3.1. Video frame sequence and frame types

For network evaluation purposes, there has been a number of approaches to prepare data forms to be transmitted through the network to examine network behavior in the state of transmitting video data. Thus, in order to do that, we explored three methods to formulate a video data: using FFMPEG platform, video trace file, and video traffic model. Some of these methods used real video streams to extract the data ready to be transmitted, others are using video streams imitating algorithms to mimic a real video data stream. At the end, the purpose is network performance evaluation.

3.1.1. FFMPEG platform

FFMPEG is a widely known tool utilized by popular platforms like Facebook, Google Chrome, and YouTube to handle videos, HTML5 video support, and convert videos [48]. In addition to that, it is an entirely open source, supports encoding/altering video/audio/image formats and has an intensely detailed set of options. FFMPEG reads contents of any number of inputs from memory then processes it according to the entered parameters or program defaults and writes the results to any number of outputs. Inputs and outputs can be computer files, pipes, network streams, grabbing devices, etc. we are concerned with transcoding process in FFMPEG, demuxers from libavformat library to read inputs and get from them packets with encoded data. Those packets are sent through the network system to be sent then to the muxer and written to the output stream. Figure 3.2. depicts how FFMPEG libraries are utilized for general and specifically for network performance evaluation.

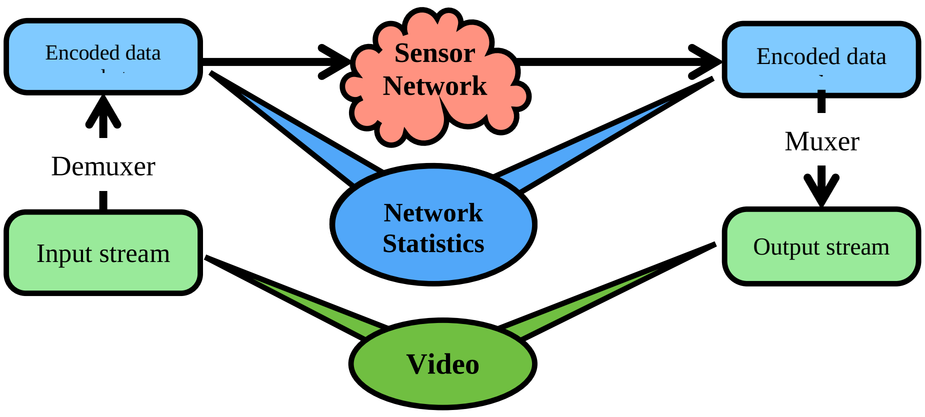
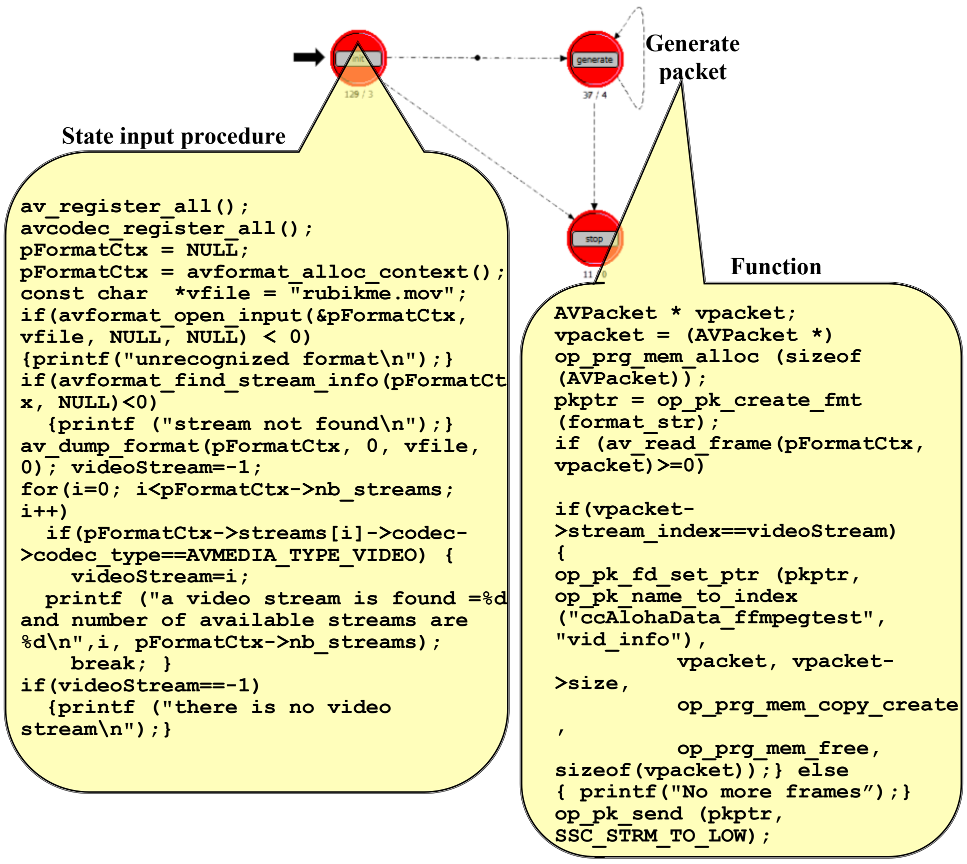


Figure 3.2. The use of FFMPEG libraries to produce a stream for evaluating the network

As it can be seen in Figure 3.2., it can be noticed that two types of statistics can be collected from the system, we called them network statistics and video statistics. Network statistics include end to end delay, throughput, packet/frame loss and jitter that reflect the performance of the network itself with the existence of such a stream, whilst video statistics include video quality measures like PSNR and SSIM that reflect the effect of the network system on the video stream itself [49].

From the three types of approaches for creating video traffic for network evaluation, only the one using FFMPEG libraries can give an idea about the effects of the network system on the video stream, i.e. its quality change.

We used FFMPEG libraries in our research work by embedding the libraries into OPNET process module codes. Figure 3.3. shows the codes and their place in the process module of the source node producing the stream from a real video file, and Figure 3.4. shows the reception side of the video stream and how it’s decoded.



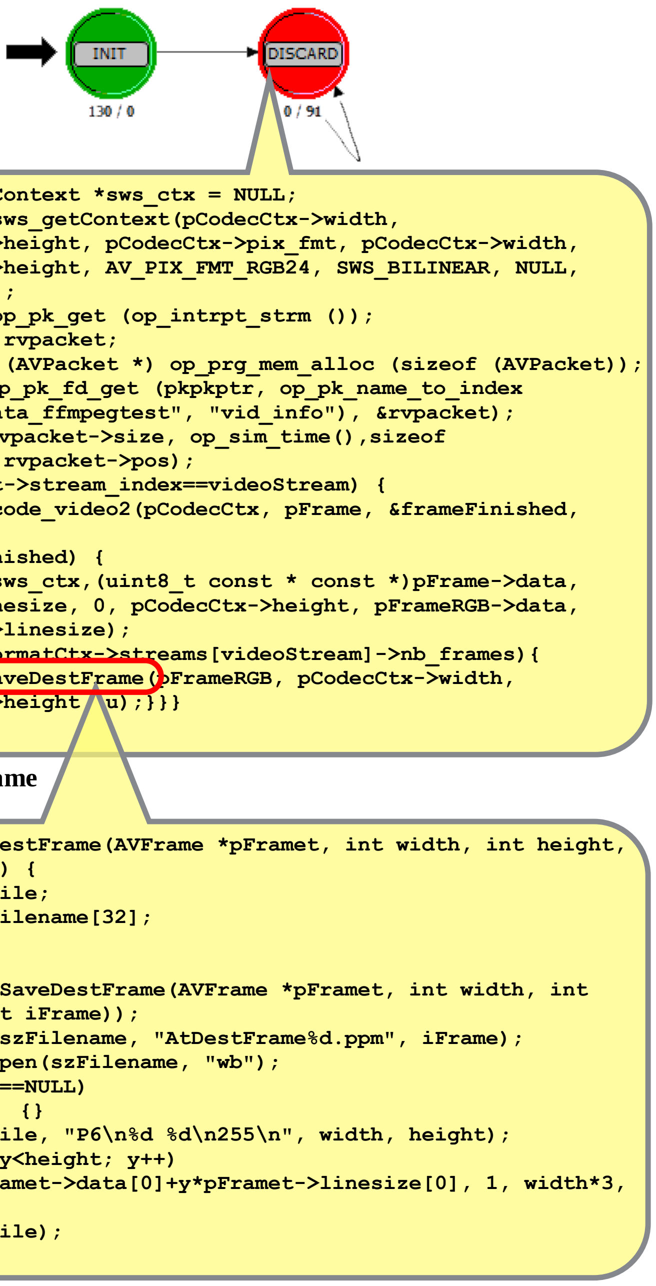
```
av_register_all();
avcodec_register_all();
pFormatCtx = NULL;
pFormatCtx = avformat_alloc_context();
const char  *vfile = "rubikme.mov";
if(avformat_open_input(&pFormatCtx,
vfile, NULL, NULL) < 0)
{printf("unrecognized format\n");}
if(avformat_find_stream_info(pFormatCt
x, NULL)<0)
  {printf ("stream not found\n");}
av_dump_format(pFormatCtx, 0, vfile,
0); videoStream=-1;
for(i=0; i<pFormatCtx->nb_streams;
i++)
  if(pFormatCtx->streams[i]->codec-
>codec_type==AVMEDIA_TYPE_VIDEO) {
    videoStream=i;
   printf ("a video stream is found =%d
and number of available streams are
%d\n",i, pFormatCtx->nb_streams);
    break; }
if(videoStream==-1)
  {printf ("there is no video
stream\n");}
```

```
AVPacket * vpacket;
vpacket = (AVPacket *)
op_prg_mem_alloc (sizeof
(AVPacket));
pkptr = op_pk_create_fmt
(format_str);
if (av_read_frame(pFormatCtx,
vpacket)>=0)

if(vpacket-
>stream_index==videoStream)
{
op_pk_fd_set_ptr (pkptr,
op_pk_name_to_index
("ccAlohaData_ffmpegtest",
"vid_info"),
         vpacket, vpacket-
>size,
         op_prg_mem_copy_create
,
         op_prg_mem_free,
sizeof(vpacket));} else
{ printf("No more frames”);}
op_pk_send (pkptr,
SSC_STRM_TO_LOW);
```

Figure 3.3. Source traffic process module using FFMPEG libraries

A simple source process module is used to develop a traffic source that produces packets with payloads of video frames taken from a video stream injected into this process. The init state would include some normal simple codes in addition to some definitions of FFMPEG libraries and opening the video stream as a file to read, then a recognizable video data will be searched, if found, the kernel would transit to generate state. In the case of finding a recognizable video data, the generate state will call packet generate function every inter-frame time (time between two consecutive frames) obtained from the stream itself according to the CODECs used in the stream. The packet generate function contains allocating memory for the video frame data to be held in there, then a pointer to this memory is sent to the lower layers using a special type packet to be a payload in the protocol packet to be received in the network sink node.



```
struct SwsContext *sws_ctx = NULL;
sws_ctx = sws_getContext(pCodecCtx->width,
pCodecCtx->height, pCodecCtx->pix_fmt, pCodecCtx->width,
pCodecCtx->height, AV_PIX_FMT_RGB24, SWS_BILINEAR, NULL,
NULL, NULL);
pkpkptr = op_pk_get (op_intrpt_strm ());
AVPacket * rvpacket;
rvpacket = (AVPacket *) op_prg_mem_alloc (sizeof (AVPacket));
          op_pk_fd_get (pkpkptr, op_pk_name_to_index
("ccAlohaData_ffmpegtest", "vid_info"), &rvpacket);
          rvpacket->size, op_sim_time(),sizeof
(AVPacket),rvpacket->pos);
if(rvpacket->stream_index==videoStream) {
avcodec_decode_video2(pCodecCtx, pFrame, &frameFinished,
rvpacket);
if(frameFinished) {
sws_scale(sws_ctx,(uint8_t const * const *)pFrame->data,
pFrame->linesize, 0, pCodecCtx->height, pFrameRGB->data,
pFrameRGB->linesize);
if(++u<=pFormatCtx->streams[videoStream]->nb_frames){
          SaveDestFrame(pFrameRGB, pCodecCtx->width,
pCodecCtx->height, u);}}}
```

```
void SaveDestFrame(AVFrame *pFramet, int width, int height,
int iFrame) {
  FILE *pFile;
  char szFilename[32];
  int  y;

FIN (void SaveDestFrame(AVFrame *pFramet, int width, int
height, int iFrame));
  sprintf(szFilename, "AtDestFrame%d.ppm", iFrame);
  pFile=fopen(szFilename, "wb");
  if(pFile==NULL)
            {}
fprintf(pFile, "P6\n%d %d\n255\n", width, height);
 for(y=0; y<height; y++)
fwrite(pFramet->data[0]+y*pFramet->linesize[0], 1, width*3,
pFile);
 fclose(pFile);
FOUT;}
```

Figure 3.4. Traffic sink using FFMPEG libraries

The sink on the other hand in the sink node will receive this packet, do some scaling then reconstruct the video frame contained in the packet using SaveDestFrame function to save video frame as an image to be viewed later for comparison with the original video stream and later for collecting some video quality measures like PSNR and SSIM.

3.1.2. Video frame size trace file

The basic concept of video frame size traces is that without the use of real video data, the system can use an array of video frame sizes with their time to create chunks of random data with sizes obtained from the array which is provided by another platform. The purpose is to test the network system with such sizes during a certain time.

In [50], Fitzek and Reisslein referred to a publicly accessible library of frame size traces of various encoded video files, which have been generated in Telecommunication Networks (TKN) group at technical university Berlin, (the trace library was accessed in 2018 at http://www2.tkn.tu-berlin.de/research/trace/trace.html).

The video frame size trace file can be produced for the same video stream file at different quality level with different resolutions and frame rate that may lead to a variety of video streams that are different in frame sizes and their associated times and those can be used to evaluate network systems with a realistic type of data as an input to the network. Figure 3.5. shows a simple idea on how these files are produced.

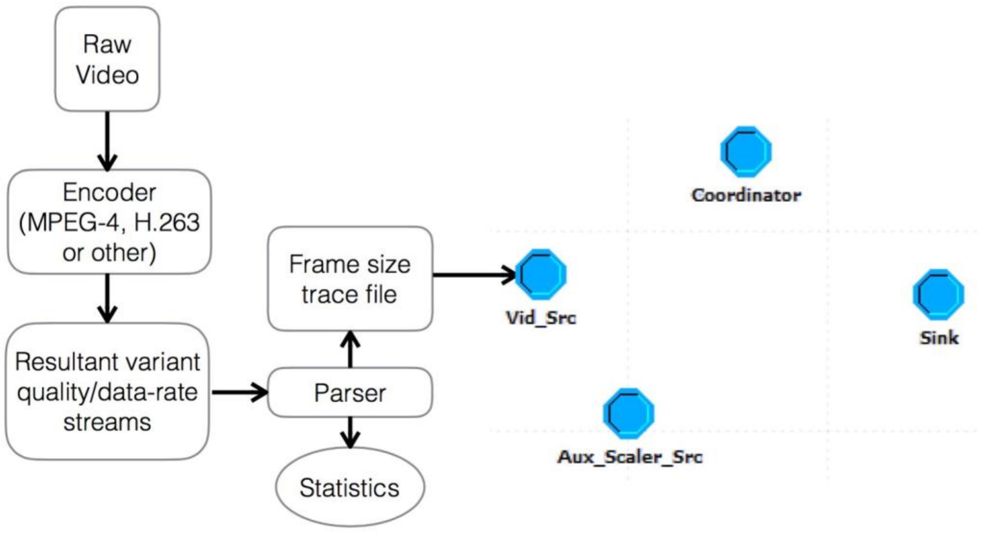
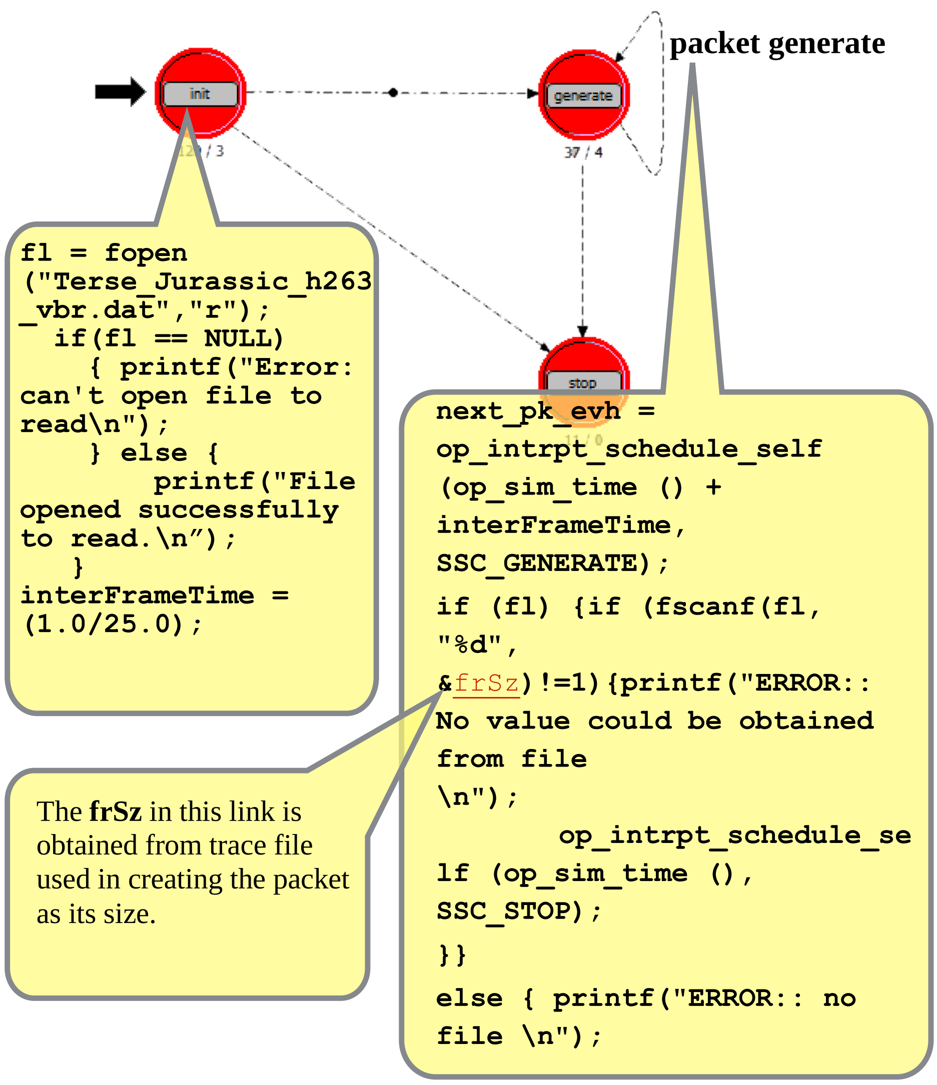


Figure 3.5. Video frame size generation and usage for network evaluation

Consequently, network system effect on video quality cannot be measured using the approach, because no real video data is used, but it can efficiently test the network framework performance and see its behavior when such stream with variable attributes transferred through it.

Figure 3.6. shows an Riverbed traffic source process module using video frame size trace file described earlier. The trace file contains frame sizes as integer numbers so the process is simple as reading from text file number by number with intervals of 1/(frames per second) parameter (which is determined by the imported trace file properties). With each time a number is read, a packet is created with its size set to the number obtained from the trace file then sent to the lower layers. When the file is finished the process simply will go to stop status probably would end the simulation.



```
fl = fopen
("Terse_Jurassic_h263
_vbr.dat","r");
  if(fl == NULL)
    { printf("Error:
can't open file to
read\n");
    } else {
        printf("File
opened successfully
to read.\n”);
   }
interFrameTime =
(1.0/25.0);
```

The **frSz** in this link is obtained from trace file used in creating the packet as its size.

```
next_pk_evh =
op_intrpt_schedule_self
(op_sim_time () +
interFrameTime,
SSC_GENERATE);
if (fl) {if (fscanf(fl,
"%d",
&frSz)!=1){printf("ERROR::
No value could be obtained
from file
\n");
        op_intrpt_schedule_se
lf (op_sim_time (),
SSC_STOP);
}}
else { printf("ERROR:: no
file \n");
```

Figure 3.6. OPNET traffic source process module using video frame size trace file

3.1.3. Video traffic imitation model

Based on the basics of video stream structures explained earlier in video traffic representation, some researcher tried to model video stream by calculating number and size of the three types of composed frames (I, P and B frames), measure their sizes and the correlation among them. an estimation of the main video stream could be imitated by decomposing the mainstream into three separated interfered steams [51].

Based on these researches, there are some attributes that define the shape of the imitated video stream such as GoP, frame rate and frames pattern. For certain values for these attributes, high bandwidth demanding video streams can be produced, and for various set of values, different video streams can be presented.

We vary those parameters values in different scenarios and examine their effects on system performance. Different stream structures are introduced for various types of frames with extremist specifications that will cause better video quality but may bring network system to fail and vice versa.

Figure 3.7. shows Riverbed process module of the video traffic model as a traffic source for the video source node module. We can see when start trigger is fired from lower layer according to the time determined by the node attributes, the traffic source process will go to mpeg_gen state that will control how I, P, and B video frames are produced in accordance with video parameters set in the nodes attributes. For each special interrupt, a different state transition is activated to produce a certain type of video frame to be sent to the lower layers.

We executed the three approaches using Riverbed simulation tool as explained earlier, and we conducted some results for comparing video streams produced by each approach. The measured parameter illustrates the required specification of the network to efficiently transmit such a stream.

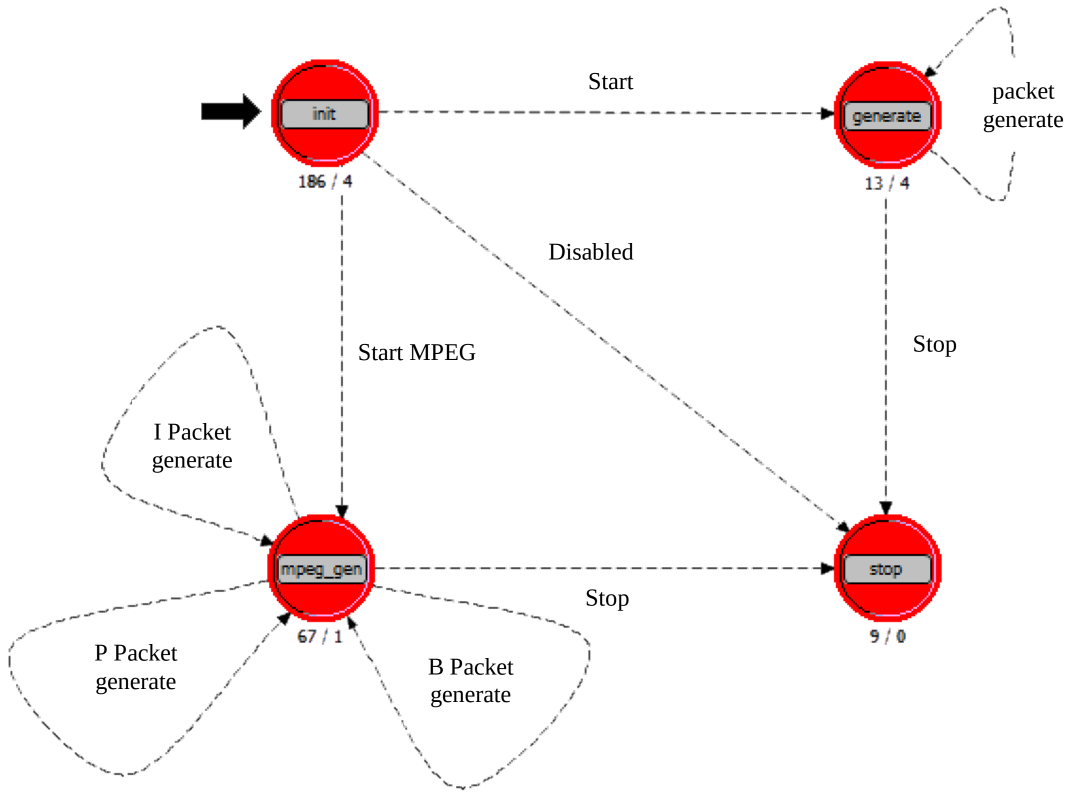


Figure 3.7. Video traffic model OPNET source process module

Figure 3.8. shows the arrival rate of the three approaches with close video attributes to show the difference among them. By observing these graphs, it can be noticed that variability in video frame size is more apparent in the video streams produced using trace file and ffmpeg libraries, whilst the video traffic model is not obvious.

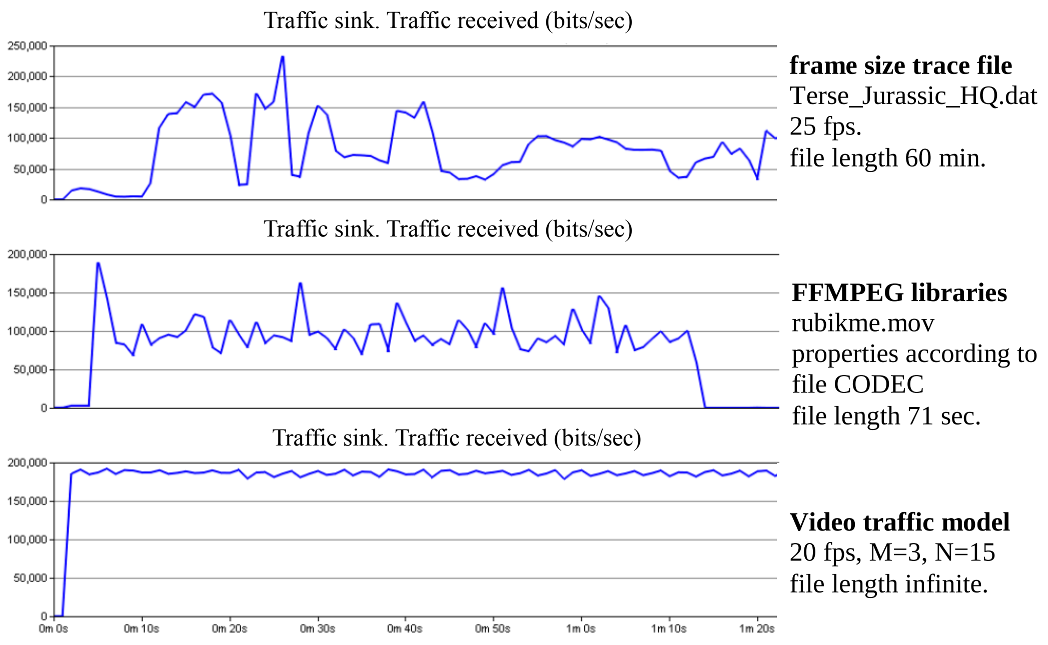


Figure 3.8. Arrival rate of three approaches of video traffic production

As a summary, FFMPEG and video file trace approaches tends to be more realistic than the video traffic model approach yet restricted to the parameters given or the traces give, while the video traffic model is more controlled by the attributes that can be set by the model given and construct different video streams to test the network system. For that reason, we used video traffic model approach for testing the traditional WSN framework in this chapter and the proposed network framework in the next chapter.

3.2. Video Streaming Over Zigbee Based Network

In the interest of putting WSN to test, we used an Riverbed model of ZigBee based WSN platform of ART-WiSe research group in Polytechnic Institute of Porto Portugal [52] The original OPNET model was modified to work with video streams, thus an MPEG2 data packets producing module was added [51]. Then the lower modules were edited to receive such packets. The video data packets are produced with a video traffic model that imitates a real video stream with the possibility of changing video parameters [53]. The video stream producer module creates frame types explained earlier with varying sizes. The lower modules segment these frames to WSN packet sizes then forwarded to lower layers for transmission[45].

3.2.1. Network topology and sequence diagram

We introduce a network structure that is depicted in Figure 3.9, which shows a coordinator and end devices as classified by the IEEE 802.15.4 standard [54]. We added different types of data production modules to end devices such as EDs that produce video streams, other EDs produce scaler data and other end devices that act only as network sink.

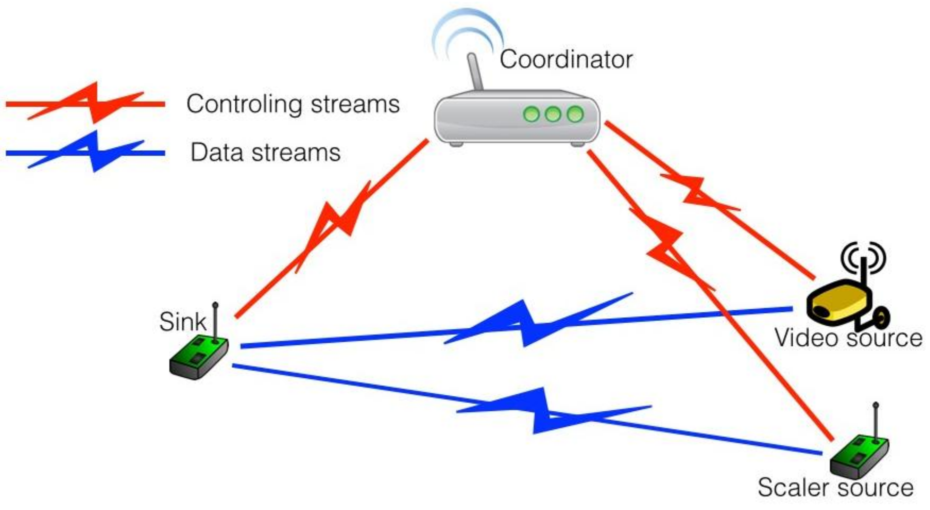


Figure 3.9. The etwork topology of the tested network

One of the functions of the coordinator is transmission synchronization using beacons and manage the network to be in a 96.8% sleep time to reserve energy. Besides that, accept transmit requests from other nodes during wake time then respond with a grant using beacons. Moreover, coordinator changes transmission timings according to the data type of the transmission requests, i.e. if video source end device requested a transmission grant the network goes to all wake up to deliver the requested transmission, whilst in case of scaler source end device transmission request, the network will keep on 96.8% sleep time.

At network establishment, the coordinator initiates the network by broadcasting a beacon regularly specified by coordinator attributes which determine the inter interval times of beacons and with parameters set by the beacon itself each end device will learn when it can compete for transmitting scaler and request transmissions.

At this state of the network, the beginning of establishment with any transmission requests, the network is considered in a sleep mode where 96.8% of the time the end devices are not transmitting/receiving any type of packets. Any changes in these parameters that would be stated by the coordinator will be conveyed to the end devices by the beacons.

The network sink at the meantime informs the coordinator about being the network sink during the 3.2% of the time that the network is awake in it. The coordinator, in turn, encapsulate this piece of information in the beacon, and accordingly, all end devices are aware of the network sink address.

As a typical scenario, one of the end devices sends a transmission request during the time where end devices can send requests. The coordinator looks into the available resources in terms of time slots, see if it’s not occupied by other transmitting end devices and grant the request accordingly by means of the beacon. Thus, the next beacon will contain the grant to the specific end device and at the same time the new network parameters that will cause the network to change transmission timings. In the case of not enough resources (time slots), the grant will be denied and the transmission will not occur [45].

At all times scaler type of data can be transmitted using the devices and the transmission requests are for video source end devices to allocate needed time slots for the requested transmission. At the end of transmission, the end device informs coordinator about transmission termination, and in turn, the coordinator sends a beacon with default network parameters and restore the network to its previous state. Figure 3.10. shows a sequence diagram of network establishment and typical scenario of the given framework.

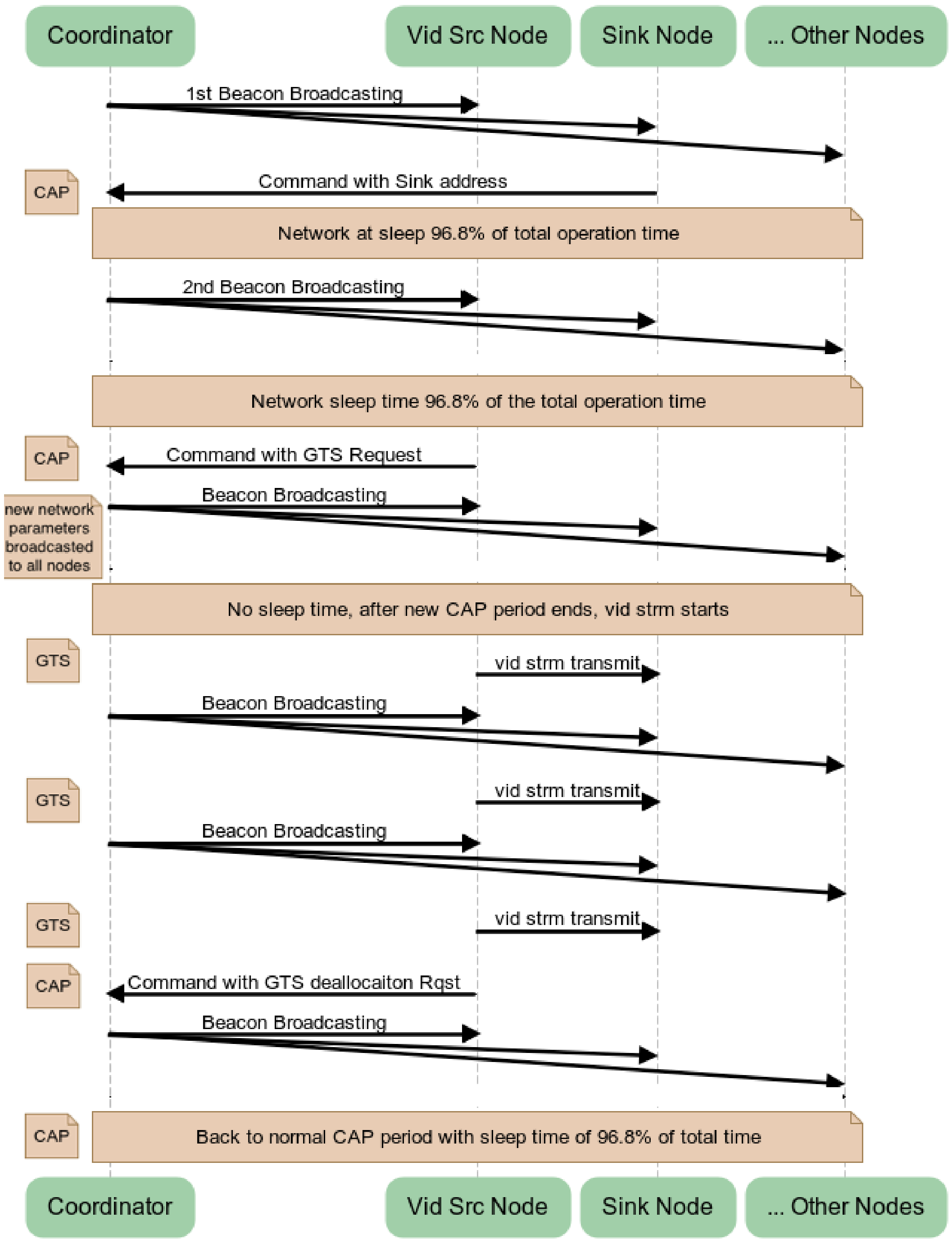

Coordinator
Vid Src Node
Sink Node
... Other Nodes
1st Beacon Broadcasting
CAP
Command with Sink address
Network at sleep 96.8% of total operation time
2nd Beacon Broadcasting
Network sleep time 96.8% of the total operation time
CAP
Command with GTS Request
new network parameters broadcasted to all nodes
Beacon Broadcasting
No sleep time, after new CAP period ends, vid strm starts
GTS
vid strm transmit
Beacon Broadcasting
GTS
vid strm transmit
Beacon Broadcasting
GTS
vid strm transmit
CAP
Command with GTS deallocaiton Rqst
Beacon Broadcasting
CAP
Back to normal CAP period with sleep time of 96.8% of total time
Coordinator
Vid Src Node
Sink Node
... Other Nodes

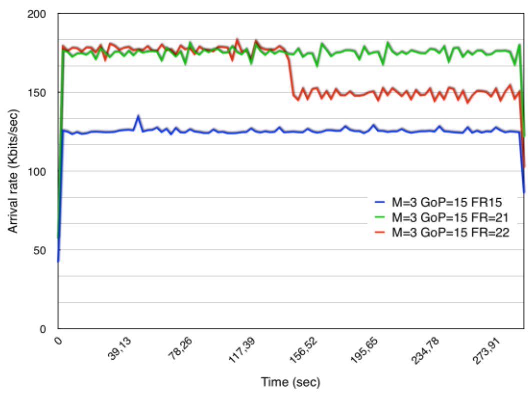
Figure 3.10. Sequence diagram of network establishment and typical scenario

3.2.2. Simulation and results

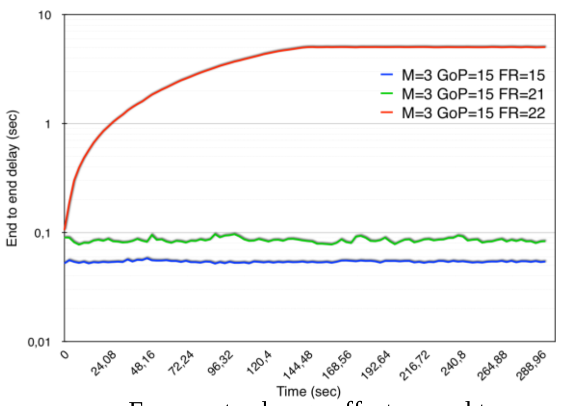
We used Riverbed to test the given framework with the parameters specified by the model. Riverbed is deemed to be a realistic tool for optimizing network frameworks. The scenarios applied in the simulation were created by setting different parameters in different situations. Network management parameters were set to defaults for ZigBee based WSN such as BO and SO to 6 and 1 in no transmission state and 3 for both of them in case of video transmission request, and a queue size of 1Mb.

The parameters that were varied for testing are video stream parameters. The frame rate was set to 15, 21 and 22 frames per second. GoP where set to 3, 4 and 15. The number of M which changes frame pattern is set to 0 or 3. The combination of M value and GoP value will decide the resultant pattern of frames like for example for GoP=3 and M=0, the resultant pattern is IPPIPPIPP.. and for GoP=15 and M=3 the resultant pattern is IBBPBBPBBPBBPBBIBBP.. and so on. Also, the sizes of I, P and B are variable also with a distribution of Log-Normal function of different arguments for each type of frame [45]. The given variety in video parameters results in extreme different video streams to be produced.

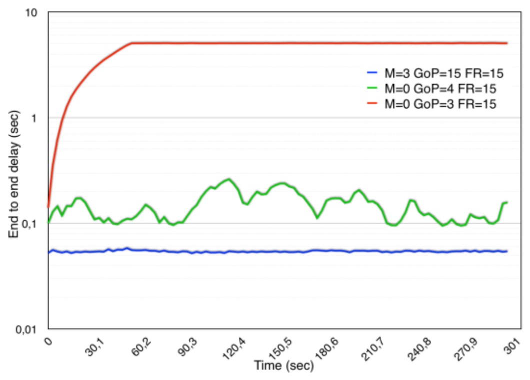
To conduct numerical result to evaluate the system, performance metrics of end to end delay and arrival rate, are used to measure how the system is reacting in such conditions. Packet loss metric is not used as buffers have enough size and the transmission medium is free space so there is no benefit from measuring this metric, yet in case of extreme video parameters, acute degradation in data delivery is perceived that yield to queue overflow and mass packet loss that can be spotted from arrival rate and end to end delay results as shown in Figure 3.11.



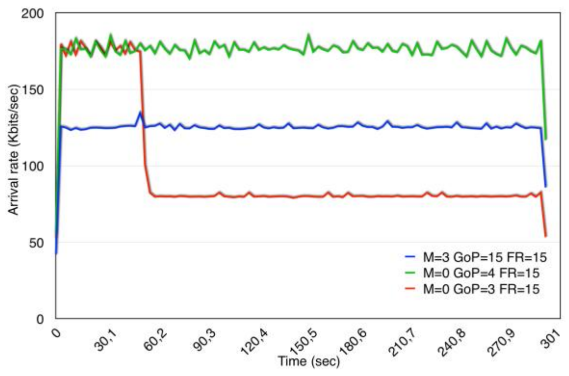
Frame rate change effect on data arrival rate at the destination



Frame rate change effect on end to end delay



GoP and M parameters change effect on end to end delay



GoP and M parameters change effect on arrival rate at destination

Figure 3.11. Numerical results for end to end delay and arrival rate for a different combination of video parameters as denoted with each graph

Other observations can be noticed from the graphs, the extreme value of frame types patterns causes faster degradation in performance than the extreme value of frame rate. The cause for this difference is that the change in frame rate results in more frames of small size types of frames (i.e. P and B) to be inserted to stream, while the change in frame type pattern leads to increase of big size frame types (i.e. I) to be inserted to stream and consequently, with certain amount of these frames the buffer starts to overflow and lose control of balancing between input and output streams.

3.3. Conclusion And What SDN Concept Can Offer

Video delivering network system was established using Zigbee based WSN. Video stream shaping parameters were varied accordingly to test the system performance under normal parameters and then under severe parameters. an overflow was observed for a certain set of parameters.

Although some functionalities were used for the sake of lower energy consumption like sleep periods management during no transmissions state of the network, there is still a need for algorithms to lower down power consumption and simplify network management, and here comes the role of SDN as a trending paradigm aiming at these issues in WSN. The next two chapters introduce the SDN solution for WSN and discuss its testbeds in both simulation platform and real practical platform.

CHAPTER 4. INTEGRATING SDN INTO WSAN

This chapter is about our main proposal for sensor networks which is an SDN-enabled wireless sensor and actuator network framework with a proposed routing discovery mechanism. In this chapter, a simulated version of the proposed platform is presented with simulation environment details and results, as a real implementation of the proposed system is introduced in the next chapter.

The sections of this chapter start with a brief introduction about how all components are combined together to form the whole system, then, an algorithm for topology discovery and clustering is described as an essential part of the system. The third section explains the key interface protocol WSANFlow and its components. How fuzzy and Dijkstra algorithms are used in the proposed routing mechanism is presented in the fourth section. Other sections are about the simulation environment parameters, scenario, results, discussion and lastly a conclusion regarding the results in this chapter.

4.1. Introduction

Transferring data using devices that utilize shared medium like wireless with topologies of cluster tree or mesh is an essential issue to consider in WSAN platforms. Aspects of considerations include power drawing rate, QoS and application-specific demands. Energy-aware algorithms for finding an optimal route from source to sink is a key challenge designing protocols for WSAN systems [3][26][24].

A trending SDN paradigm is taken as a resolution for WSAN to tackle these issues and put forward a new WSAN protocol for the new applications arose recently [1][55].

Thus, we present in this chapter a new protocol including a routing decision module that is based on the SDN paradigm with the usage of fuzzy Dijkstra's algorithm. Development of the protocol model is done using Riverbed Modeler network optimization tool along with performance evaluation in terms of specific aspects. The key parts of the proposed protocol are SDNC, SDN-enabled ED, and WSANFlow. SDNC is the controller node that holds the knowledge needed to manage the network and algorithms that use this knowledge to make requested/required decisions. SDN-enabled EDs are the data-plane devices that are instructed by SDNC to carry out their specified tasks accordingly. The last one is the interface protocol (WSANFlow) that maintain a reliable mutual understanding between SDNC and SDN-enabled EDs by defining certain rules, packets formats and fields, control messages and managing flags that can be comprehended by both SDNC and SDN-enabled EDs [31].

4.2. Topology Discovery And Clustering

In order to get neighbors list that is an important piece of information in the knowledge base of the SDNC, which is used for grasping the big picture of the whole network. The neighbor list consists of list neighboring devices of each device and their associated link quality in terms of SNR values. TDM is the mechanism that facilitates the creation of neighboring lists in EDs by designating clusters in the network in a specific manner and initiating certain procedures that result in each ED to be aware of its neighboring devices. Consequently, these lists are transmitted to SDNC in procedural steps determined by SDNC in accordance with the WSANFlow protocol for managing such transmissions.

TDM is initiated by SDNC by setting a flag in the beacon so that all EDs broadcasts their own TDM beacon to be seen by other EDs and store address and SNR value associated with the incoming TDM beacon. Figure 4.1. shows a flowchart of TDM.

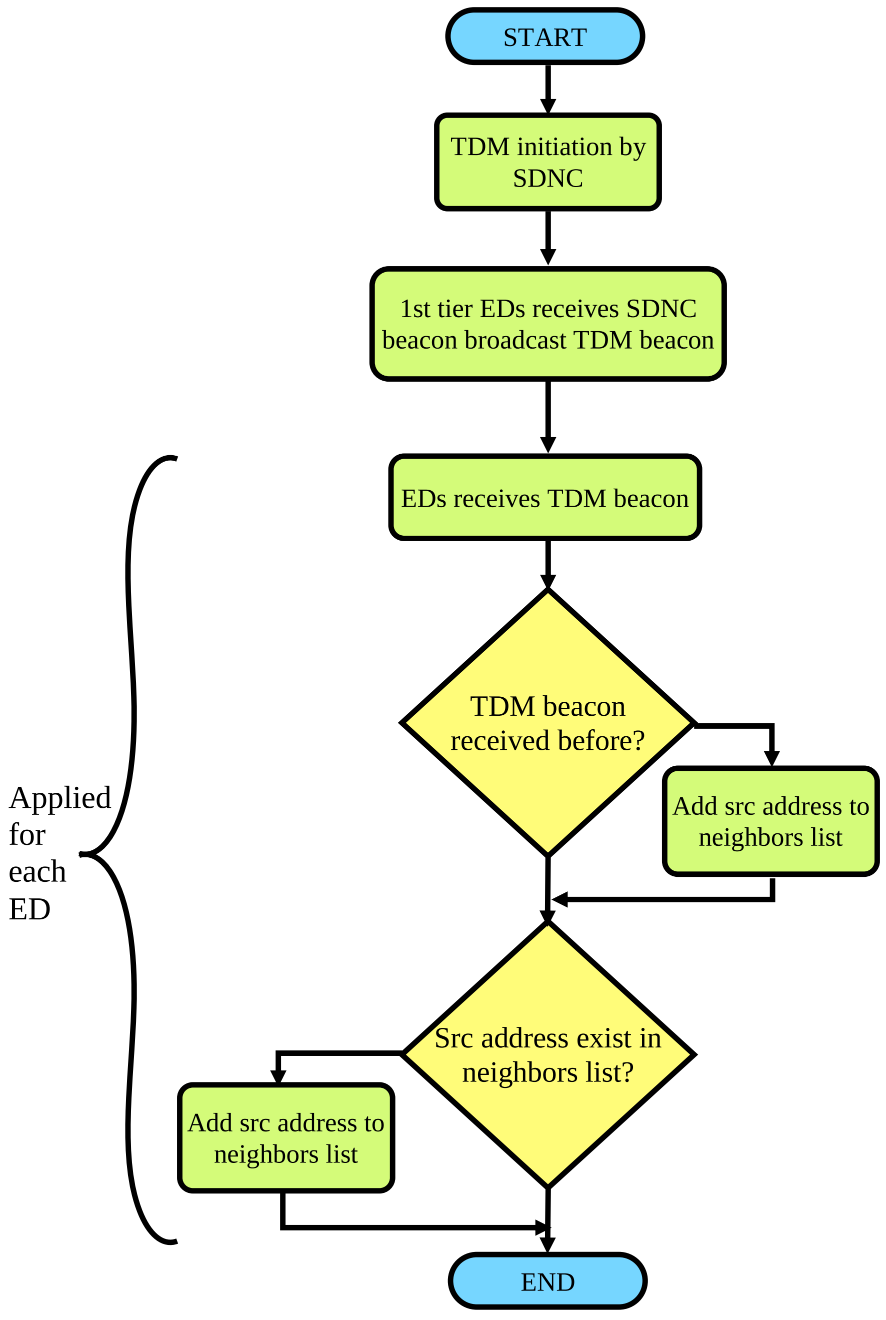

START
TDM initiation by SDNC
1st tier EDs receives SDNC beacon broadcast TDM beacon
EDs receives TDM beacon
TDM beacon received before?
Add src address to neighbors list
Src address exist in neighbors list?
Add src address to neighbors list
END
Applied for each ED

Figure 4.1. TDM flow chart

TDM includes clustering of EDs which leads to idle periods for a number of devices regularly for energy consumption purposes and for allowing other interfering clusters' EDs to communicate and gather status information such as neighbors lists within the scope of TDM. Idle periods and cluster heads are set and randomly chosen by SDNC using neighbors list information received according to a number of child devices a certain ED has. The child device is an ED that reaches the SDNC only through another ED that is considered its parent, which is chosen as a cluster head for another child device/s. Figure 4.2. depicts a sequence diagram of establishing the network in our proposed framework and it includes mostly the TDM procedures.

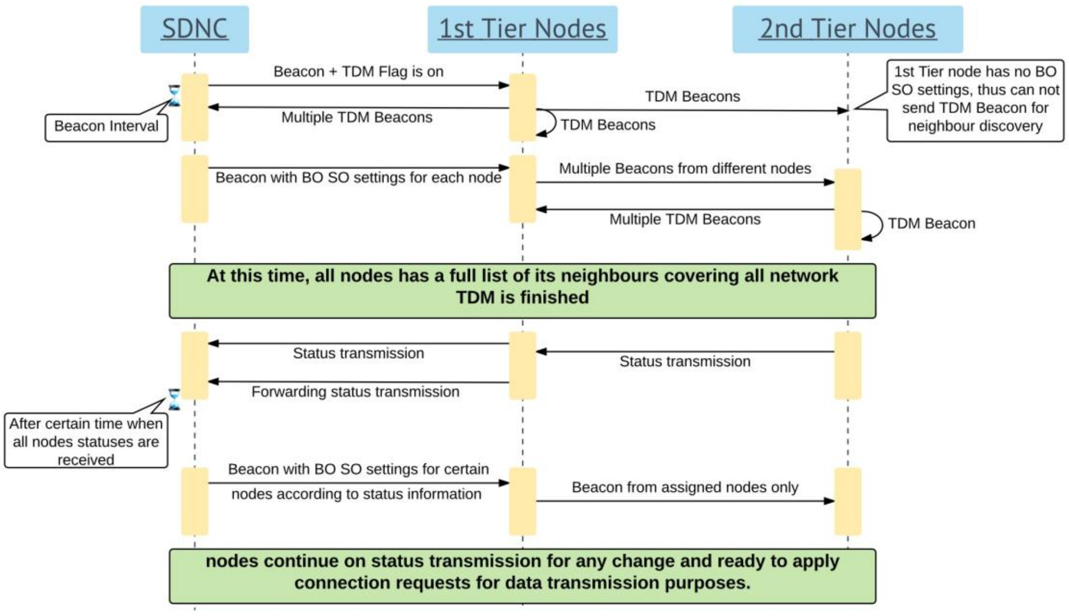

SDNC
1st Tier Nodes
2nd Tier Nodes
Beacon + TDM Flag is on
Multiple TDM Beacons
Beacon Interval
TDM Beacons
TDM Beacons
1st Tier node has no BO SO settings, thus can not send TDM Beacon for neighbour discovery
Beacon with BO SO settings for each node
Multiple Beacons from different nodes
Multiple TDM Beacons
TDM Beacon
At this time, all nodes has a full list of its neighbours covering all network TDM is finished
Status transmission
Status transmission
Forwarding status transmission
After certain time when all nodes statuses are received
Beacon with BO SO settings for certain nodes according to status information
Beacon from assigned nodes only
nodes continue on status transmission for any change and ready to apply connection requests for data transmission purposes.

Figure 4.2. Network establishment sequence diagram

According to status transmission timers of each node, SDNC will be aware of topology structure in a while. In a certain time, SDNC will be aware of that there are nodes with no children node so no need for them to have their own beacon so they are being assigned to not defined except for parent address stay as it is.

After completing the previous steps, the network would be as shown in Figure 4.3. that shows a typical topology with parameters of SO, BO, Timer, and parent address that are all set by default to not defined and then during simulation time were set to their shown values by SDNC according to the network need. If the network suffers from some types of changes, SDNC can easily adapt to the changes and modify those parameters to work energy and time efficiently.

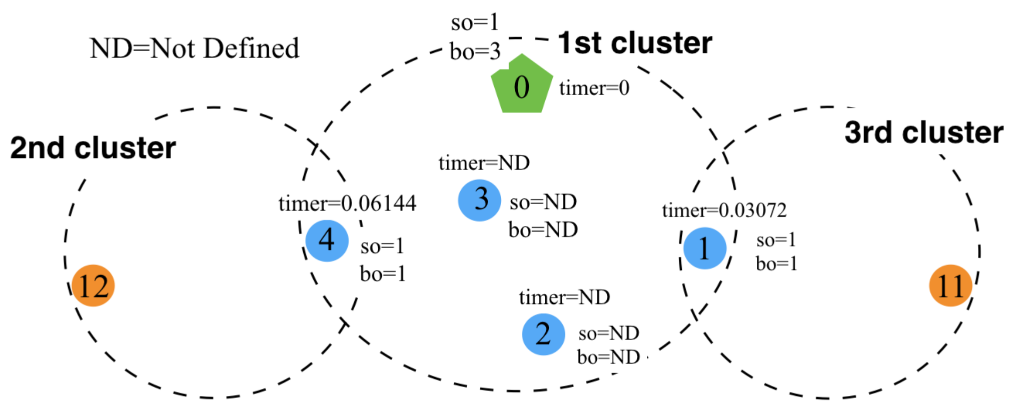


Figure 4.3. Topology with clusters and parameters assignments

4.3. SDN-oriented WSAN And WSANFlow Interface Protocol

This section put forward the proposed framework with a description of its parts and how they interact to produce the demanded performance of the whole network system. The platform consists of two kinds of network devices: SDNC and ED, and a communication protocol that governs their interaction, referred to as WSANFlow. The following sub-sections explain those parts individually with figures and tables. Also, some specific tasks included in the WSANFlow protocol will be explained such as the Network Intelligence role and Flow tables layout and formats.

4.3.1. SDN controller

The architecture of WSANFlow protocol is placed above a lower layer of IEEE 802.15.4 MAC layer, which is known standardized datalink and physical layer designed for sensor networks [54]. The main modules of SDNC are Topology discovery, Network Intel and knowledge base. The topology discovery includes codes for initiating and managing TDM procedures stated in earlier sections. This unit uses information stored in the knowledge base of statuses like neighbors lists and so to make decisions regarding the topology of the network such as setting cluster heads and setting their timings. Figure 4.4. illustrates the protocol stack of WSANFlow in SDNC.

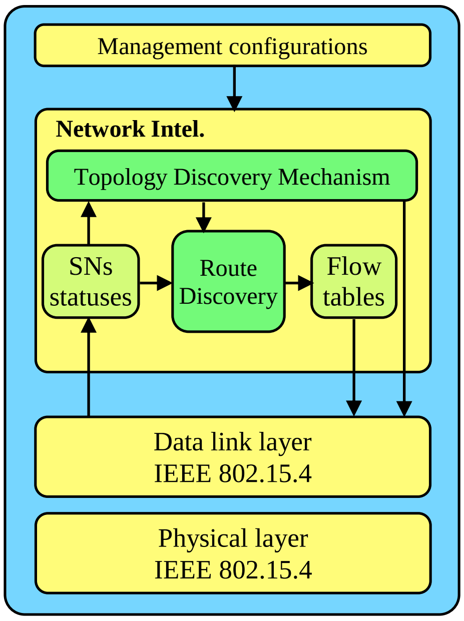

Management configurations
Network Intel.
Topology Discovery Mechanism
SNs statuses
Route Discovery
Flow tables
Data link layer IEEE 802.15.4
Physical layer IEEE 802.15.4

Figure 4.4. The protocol stack of WSANFlow in SDNC

The Network Intel consists of algorithms and procedures for making route requests decisions using updated knowledge base information. It finds the best route for the incoming request according to EDs status information stored in knowledge base then based on the resultant route, it creates appropriate flow tables for the related EDs to be sent to them accordingly. Network Intel role in the protocol and flow table creation and formats will be explained separately later in different sections.

4.3.2. SDN-oriented end devices EDs

The EDs layers protocols were designed and adjusted so that it would comply with the rules of the WSANFlow protocol for communicating with SDNC and among each others. The key modules of the SDN enabled ED is shown in Figure 4.5. The most important parts are the status collector, flow tables and the packet switch. The status collector keeps the updated values of the ED attributes such as battery level and neighbor list so that it would be sent to SDNC regularly.

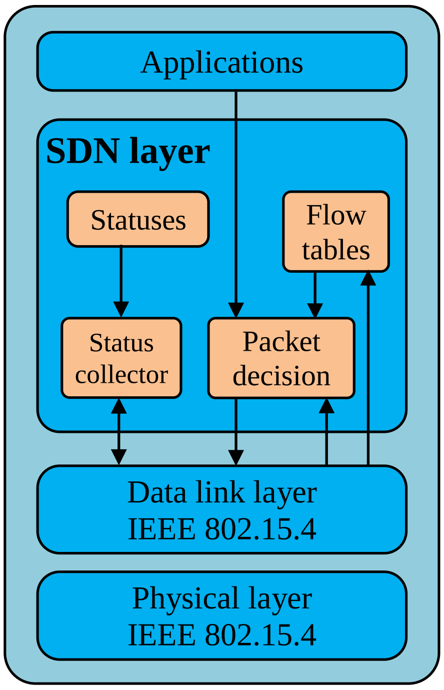


Figure 4.5. SDN-enabled ED protocol stack

Upon requesting a transmission grant from SDNC from one of the EDs, flow tables are distributed to certain individual EDs that are kept in the flow tables module in the ED. Packet switching module is consulted for every incoming data packet, which in turn looks for instructions in the flow table module regarding the incoming packet. It modifies the packet's header according to the related flow entry, then sends it to the lower/upper layer accordingly.

4.3.3. Network intelligence role

When an ED intends to transmit data packets, it sends a transmission request to SDNC. In SDNC, the Network Intel module receives this request then checks its available network resources. At this time, the SDNC is supposed to be fully knowledgeable about the network and its EDs statuses in a way it would make it uses some certain algorithms to come up with the most proper decision for a route designated for the current transmission request. Network Intel uses Dijkstra's algorithm to find the optimal route, uses link’s quality as the cost to execute the algorithm. This information is updated periodically by TDM and status transmission procedures carried out by the collaboration between SDNC and EDs.

For a more efficient performance of the network, another attribute is added to Network Intel module for more appropriate calculation of the cost value that enters the Dijkstra’s algorithm which is the ED’s battery level. A popular decision-making method is utilized to extract an accurate value for the cost for each link from the link quality and ED’s battery level, this method is fuzzy logic. The output of the Dijkstra’s algorithm will be a network state aware and power efficient decision of a route for the requested transmission. Flow tables creation comes after this step. These flow tables are then disseminated to route-involved EDs to start to transmit, receive or forward packets related to the requested transmission. Figure 4.6. illustrates the Network Intel role in the system with a flow chart.

4.3.4. Flow tables layout and dissemination

SDNC sends individual flow entries to their individual EDs via beacon to be stored in flow tables module in ED. The ED that sent the transmission request at the first place will trigger the application layer to start the data packets transmission.

The packet switch module in ED whether it’s the source ED in the route or an intermediate ED or the sink one will be called upon every incoming packet from upper/application or lower/MAC layers. The packet switch will extract the source address from the packet header to look for its associated flow entry. Once it’s found, it will go ahead in the flow entry to read the Action field of the flow entry to apply the instruction there. When the Action field is forward, the packet switch module will modify the packet’s header and change the destination address to the value taken from the Next hop field of the flow entry. For EDs that are chosen to be cluster heads, the packet switch should read the field of Cluster dir, which will decide the direction of the current packet, 0 is child cluster direction, 1 is parent cluster direction.

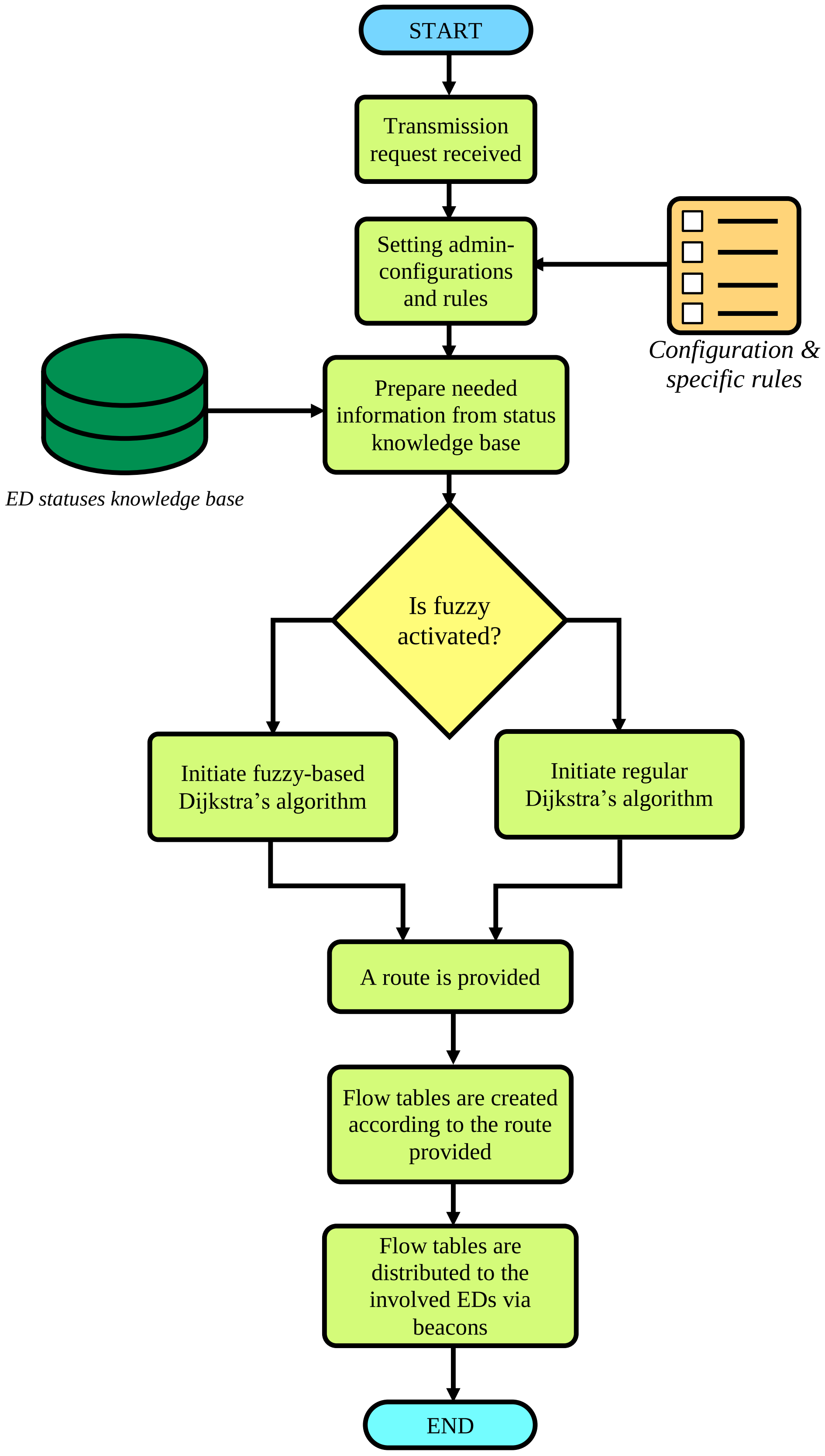

START
Transmission request received
Setting admin-configurations and rules
Configuration & specific rules
Prepare needed information from status knowledge base
ED statuses knowledge base
Is fuzzy activated?
Initiate fuzzy-based Dijkstra's algorithm
Initiate regular Dijkstra's algorithm
A route is provided
Flow tables are created according to the route provided
Flow tables are distributed to the involved EDs via beacons
END

Figure 4.6. Network Intel role flow chart

4.4. Fuzzy Model And The Dijkstra Algorithm

The task assigned for the fuzzy-logic unit in the Network Intel module in SDNC is to produce a value for each link cost in the network. To achieve that, we use fuzzy-based model, put link quality and ED's battery level as an input, the fuzzy model gives the cost as an output after a number of procedures in the fuzzy logic model. Briefly, the proposed fuzzy model does fuzzification for the input value then inference then defuzzification. During that, it converts the given crisp values for the input to fuzzy values using membership functions such as trapezoid and triangular. Then utilizes some predefined IF-THEN rule base to produce a linguistic set of the fuzzy values. Lastly, it uses the Mamdani model for defuzzification and the Center of Gravity CoG method to extract the final result of the fuzzy model which is the crisp value of the output cost [56]. Figure 4.7. depicts the fuzzy-logic model and its main parts.

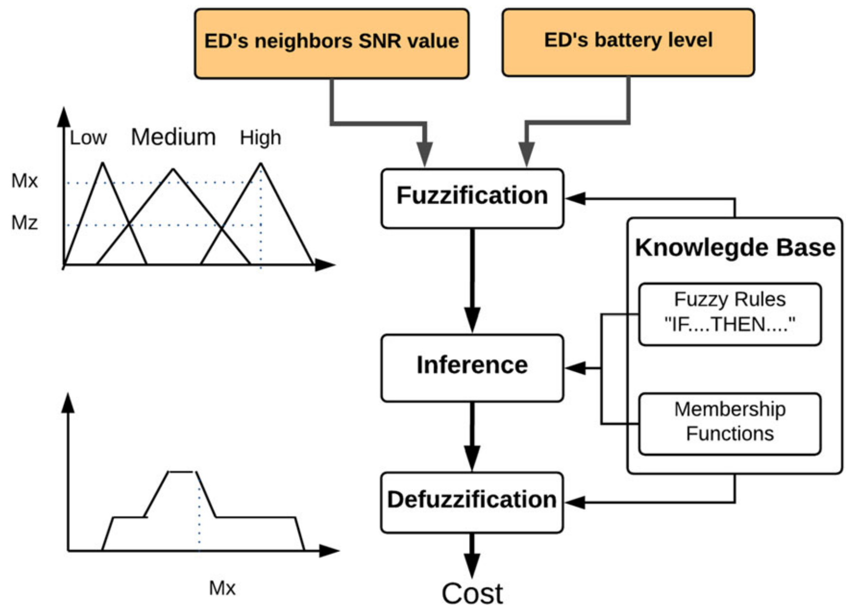


Figure 4.7. Block diagram of the given fuzzy-logic model

Dijkstra's algorithm is a recognized method for getting the shortest route in a number of distributed nodes, so it uses the distances as the input. In our proposed platform, the neighbor list in the knowledge base is used, and the distance between each two ED's is replaced with the cost which is calculated via the fuzzy model mentioned earlier.

The Dijkstra's algorithm and relatively the fuzzy model are triggered each time a transmission request is received by the SDNC, besides that, when an updated ED battery level of a below threshold value would trigger a new route discovery procedure, it consequently triggers Dijkstra's and fuzzy models.

4.5. Simulation: Parameters And System Models

We had two articles published in prestigious journals [31][56]. In this section, we would like to recall the two simulation demos we worked on, in the two articles.

The two simulation demos were carried out in the scope of this research. One of them includes a topology of five devices including the SDNC, while the later one achieved with near to 50 devices. Both of them executed in Riverbed modeler. The second one is an upgraded version of the first one with the following key:

1. The utilization of multi-input data processing algorithms like fuzzy logic coupled with Dijkstra's approach to make a network management decision.

2. A sophisticated topology discovery mechanism was used for precise ED's neighbor list and link qualities.

3. Clustering was achieved using a dynamic parameters assignment using the neighbor lists information stored in the SDNC. i.e. different clustering patterns would be executed for different neighbors lists of the network.

4.5.1. Demo number one

The parameters of the first simulation demo are shown in Table 4.2. It is categorized as general network settings, IEEE 802.15.4 settings, and data traffic parameters.

Table 4.1. Simulation parameters for demo number one

General network settings		IEEE 802.15.4 settings	
Data rate (kbps)	250	Default BO, SO (no transmission request network state)	BO = 6, SO = 1
Frequency band (MHz)	2400	BO, SO (in response to transmission request)	BO = SO = 1 to 7
Modulation mode	QPSK	Single slot duration (s)	$60 * 2^{SO} * 4/25{,}000$
Sensor node transmit power (mW)	2	Beacon Interval or Superframe Duration (s)	$960 * 2^{SO \text{ or } BO} * 4/25{,}000$
ED to ED distances average (m)	750	Addressing	Auto assign (each ED is given a unique address by a special process)
Received power threshold (dB)	− 95	Buffer size (KB)	122
Initial energy (J)	34,560		
Injected/inserted data traffic parameters			
	Light load	Heavy load	Heavier load
Packet size (bit)	Uniform (35–280)	lognormal (4500, 24,000)	I: lognormal (23,000, 240,000) P: lognormal (8000, 16,000) B: lognormal (7000, 24,000) Pattern: IBBPBBPBB
Inter-arrival time (s)	0.1	0.2	0.1111111
Transmission duration (min)	10	10	10

The topology of this demo is depicted in Figure 4.8, which shows 4 devices excluding SDNC. As a preliminary model, the control plain links are considered single hop, i.e. SDNC communicates with EDs in a star topology. while the data plain links are a multi hop, which means, there can be a multi-hop route from source to destination.

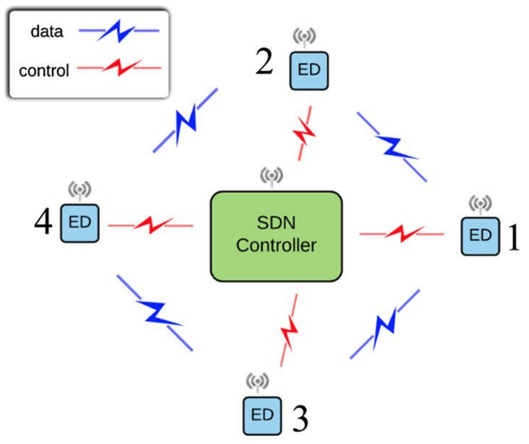


Figure 4.8. Network topology for demo number one

Our proposed platform is achieved in Riverbed modeler. Thus, some of the important node models and process models are shown below in Figure 4.9.

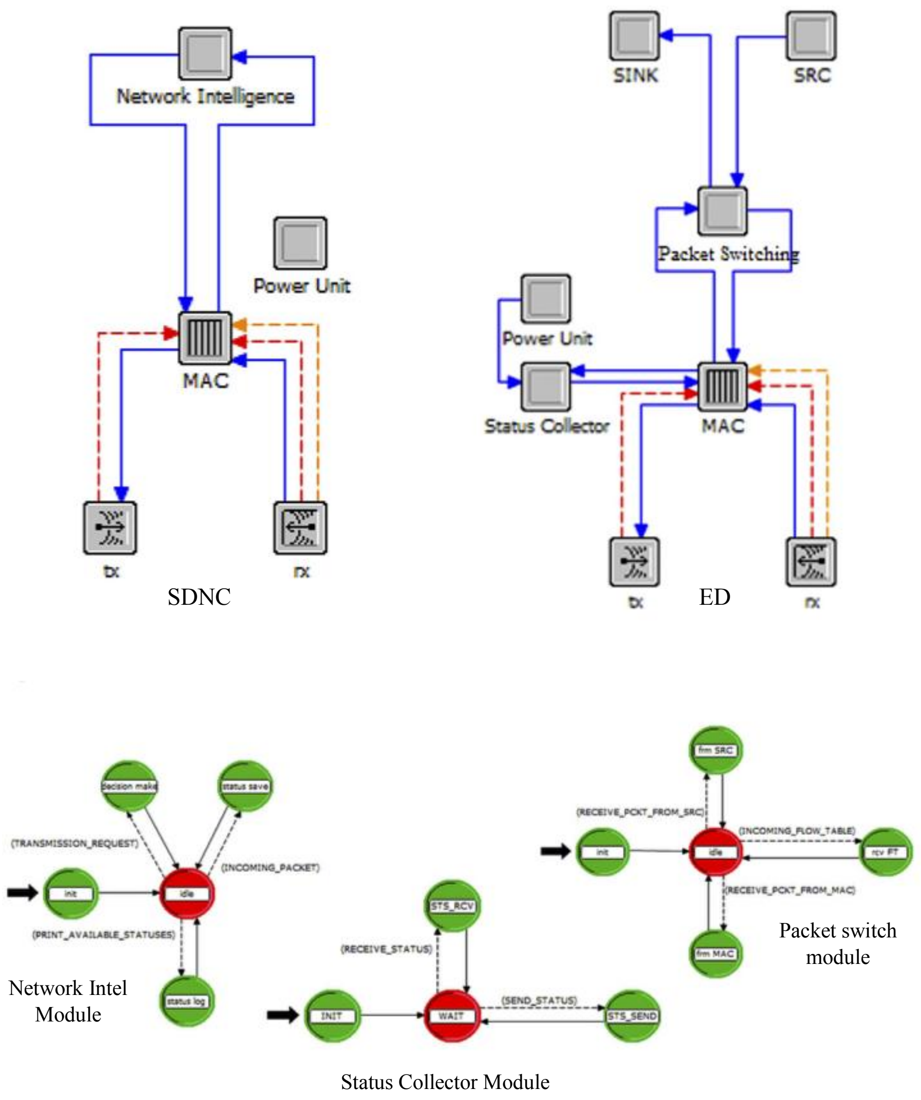


Figure 4.9. Riverbed node models and process models

The flow tables sample of a typical transmission request of the system in demo number one are shown in Table 4.2. These flow tables are created in SDNC in Network Intel module in accordance with the route created as a response to the current transmission request, then sent to the involved EDs, where it will be stored in the packet switch module inside the ED. It can be referred to the topology of demo number one mentioned above.

Table 4.2. Entries for the flow tables used packet switch module in demo one

ED	Matching rule		Action		Statistics
	Comparator	Source address	Forward/drop	Next ED	Number of packets
1	‘=’	1	Forward	2	34
	‘≠’	1	Drop	–	81
2	‘=’	1	Forward	4	33
	‘≠’	1	Drop	–	94
3	‘=’	1	Drop	–	32
	‘≠’	1	Drop	–	81
4	‘=’	2	Forward	4	33
3	‘=’	2	Drop	–	94

4.5.2. Demo number two

In the second demonstration, three scenarios were executed. The first two scenarios are the proposed platform with the use of fuzzy based route discovery and regular Dijkstra’s (without fuzzy). The third one is a traditional similar platform which is a ZigBee based platform. The aim of that is to emphasize the effect of a fuzzy algorithm usage and further show the general proposed system performance compared to a traditional one.

We developed our system to be multi-hop layout in the control plane, i.e. how the SDNC and ED’s are communicated. The topology of the second demo is shown in Figure 4.10.

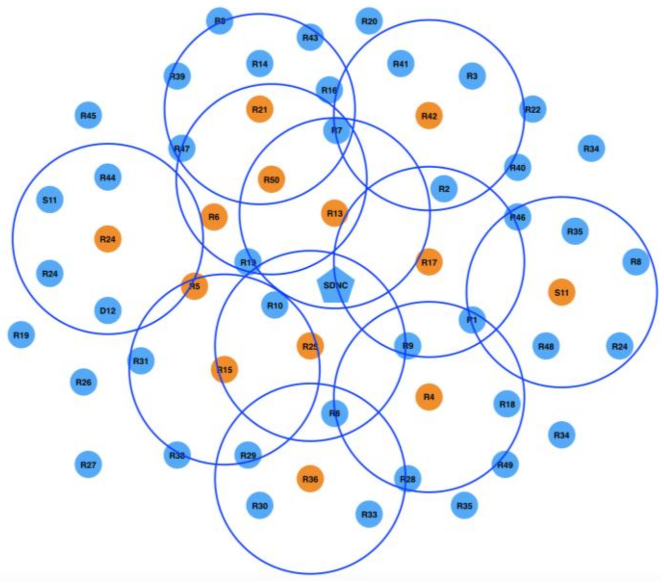


Figure 4.10. The network topology of the proposed system in demo 2

The simulation parameters for this demo is shown in Table 4.3. We used up to 50 nodes for different scenarios. The initial energy of the EDs is 5 joules, which is low level to demonstrate the network in a low level of power to get the number of nodes dying and compare them in different scenarios. Also, different scenarios for a different number of simultaneous transmission requests denoted as applications in the table.

Table 4.3. Simulation parameters for demo number two

Items	**Name**	**Value**
Network topology	Number of end devices	10, 20, 30, 40, 50
	Network coverage area	300 m × 300 m
	SDNC location (*x*, *y*)	(150, 150)
	Simulation time	10 800 s
	Max. no of EDs in cluster	15
	Path reestablishment threshold	2 J
Device settings for both the proposal and ZigBee-based WSAN	Data rate	250 kbps
	ED status transmission period	25 s
	Initial energy	5 J
	Channel model	Free-space propagation model (LoS)
	Power threshold	−76 dBm (80 mW)
Battery parameters (Micaz mode) for both the proposal and ZigBee-based WSAN	Transmission mode (0 dBm)	17.4 mA
	Receive mode	27.7 mA
	Idle mode	35 μA
	Sleep mode	16 μA
Application 1 (Src. 11 in Figure 8)	Start time	50 s
	Packet payload size	30 bytes
	Packet interarrival time	2 s[a]
Application 2 (R 34 in Figure 8)	Start time	200 s
	Packet payload size	30 bytes
	Packet interarrival time	3 s[a]
Application 3 (R 43 in Figure 8)	Start time	300 s
	Packet payload size	30 bytes
	Packet inter arrival time	5 s[a]
Application 4 (R 36 in Figure 8)	Start time	400 s
	Packet payload size	30 bytes
	Packet inter arrival time	4 s[a]
ZigBee-based WSAN	ZC beacon configurations	BO = 4, SO = 1
	Tree routing configurations	Lm = 2, Rm = 4, Cm = 4
	Number of end devices	50
	Network coverage area	300 m × 300 m
	ZC location (*x*, *y*)	(150, 150)
	Simulation time	10 800 s
	Initial energy	5 J
Application 1 (Src. 2)	Start time	50 s

SDNC and ED node models that were developed in Riverbed modeler are shown in Figure 4.11., beside some process models which play some important roles in the system like topology discovery module, the packet switch and Network Intel modules.

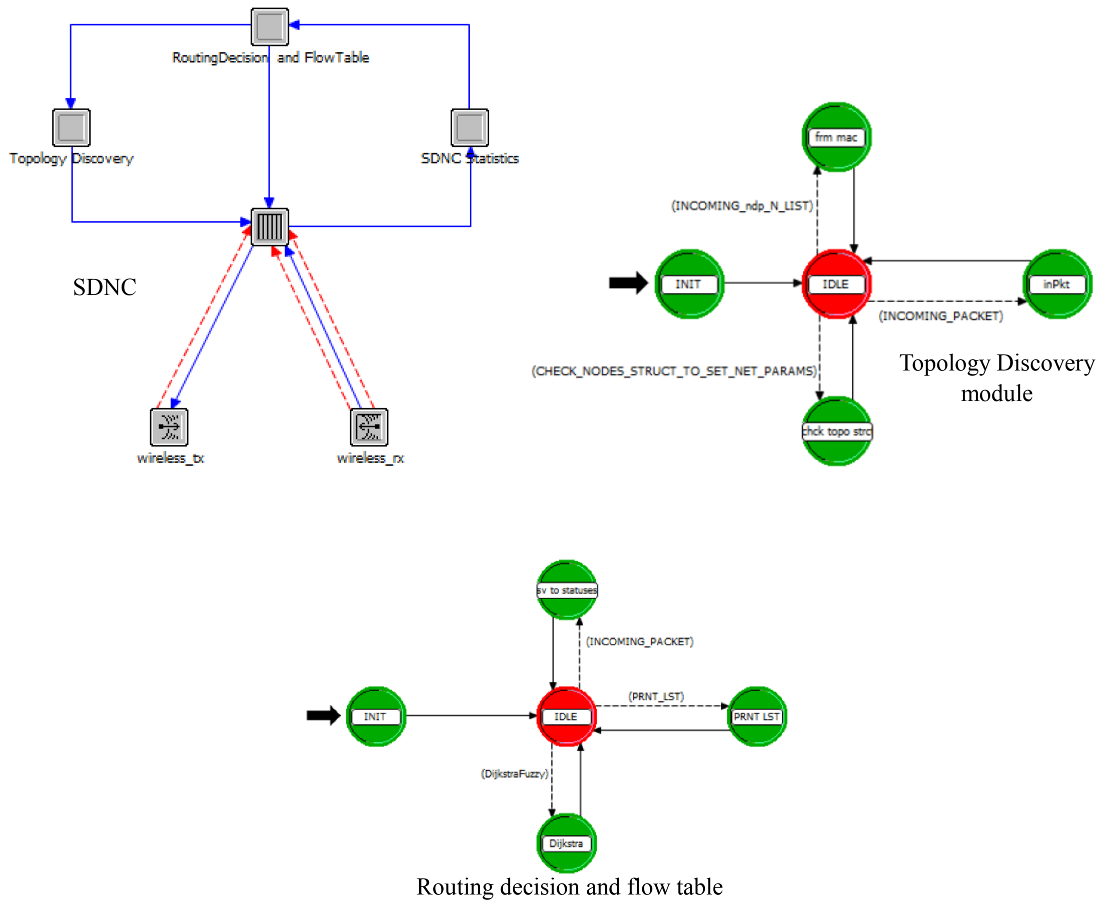


Figure 4.11. Riverbed node models and process models for demo number two

For the flow tables in demo two, application id field is added to the flow entries, in addition to cluster dir field which is needed a field for EDs that belongs to two interfered clusters to forward the packet to the correct cluster. Flow tables of different applications with the demonstrated transmission requests are shown in Table 4.4. These flow entries are created upon requests from EDs as shown in the network topology of demo 2 shown earlier in Figure 4.10.

Table 4.4. Flow tables of a demonstrated requests in demo two

		Matching rule		Action			Statistics
Application ID	Device address	Comparator	Src address match rule	Forward/Drop	Next hop	Cluster dir	Number of packets
1	13	"="	13	Forward	2	0	34
1	4	"≠"	1	Drop	-	-	81
2	5	"="	1	Forward	4	0	33
2	4	"≠"	1	Drop	-	-	94
3	6	"="	7	Forward	4	0	32
3	15	"≠"	7	Drop	-	-	81
2	5	"="	2	Modify	8	0	65

4.6. Network Performance Evaluation: Results And Discussion

4.6.1. Demo number one

The numerical results shown in this subsection are for the proposed system in its preliminary state. They are compared with the results of a traditional ZigBee-based WSAN system that uses a tree routing algorithm with similar simulation parameters [31].

We tested the proposed platform under different network and load settings. Figure 4.12. show numerical results of three different metrics: throughput, energy consumption and end to end delay. As shown in the graph, those results were fetched during simulation time by re-running the simulation several times by changing parameters of SO=BO, as shown in simulation parameters table of demo 1, for different transmission and sleeping timings. Moreover, different load parameters are changed during simulations, categorized as light load and heavy loads as shown in the simulation parameters table of demo 1 in Table 4.1. Heavy loads imitate a multimedia stream. Other simulation for a different system, which is ZigBee-based, was carried out and its results were collected also several times with the change of SO=BO to compare with our proposed one.

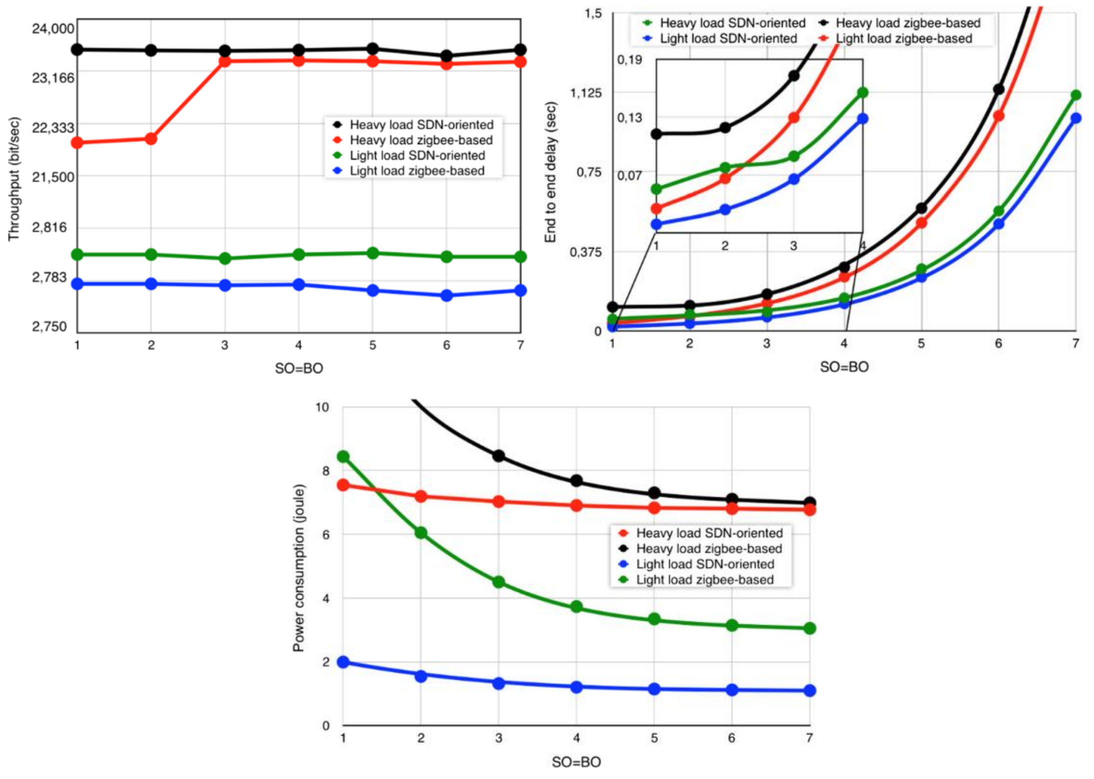


Figure 4.12. Simulation results for Demo 1 for different systems and different metrics and parameters

Looking at the throughput results of Figure 4.12., in spite the fact that these results show a very small between the two systems in both light and heavy loads, SDN-based system endures a little more than the ZigBee-based one in case of SO=BO=1 and 2.

If we observe end to end delay results of the same group of results, the average end to end delay value is noticed degrading while increasing SO=BO value. The justification for this is time slot durations getting long, which makes EDs waits longer than smaller values of SO=BO, which may cause buffers overflows and starts dropping packets. Yet SDN-based results are observed to be tolerating to these severe parameters better than the ZigBee-based counterpart.

The most important metric to check out is the energy consumption rate. The average power consumption of the routing device is shown in Figure 4.12. The routing devices consume more power than other devices since they forward packets (receive/transmit). It’s obvious from energy consumption results that the proposed SDN-based system outperforms the ZigBee-based one in both light and heavy loads. The power consumption of a heavy load traffic with SO=BO=7 is observed to be close to the same in case of SDN-based and ZigBee-based. This can be justified by knowing that sleeping periods of the proposed SDN-based system was removed for better usage of the available resources, while the ZigBee-based system is using sleep periods and they increase a lot by increasing SO=BO values. But, this causes a severe bad end to end delay results as shown in end to end delay results in Figure 4.12. of demo 1.

For a better understanding of how our proposed SDN-based system perform compared with a ZigBee-based system, Figure 4.13. is showing the average end to end delay for different loads (as shown in Table 4.1. simulation parameters of demo 1) in the case of SO=BO=3.

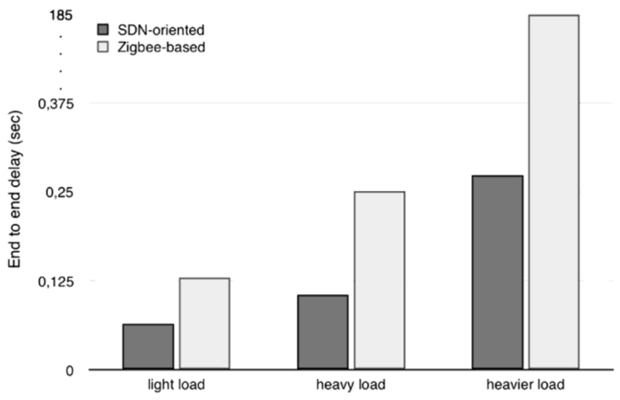


Figure 4.13. Average end to end delay for different loads when SO=BO=3 for demo 1

4.6.2. Demo number two

The results of a developed version of the proposed system from the preliminary version of demo 1 are observed in this sub-section with comments on them. We will concentrate on power consumption here as it is considered the main contribution of the research. The metric used to show how our proposed system performs against consuming power is the time when the first ED dies and the number of dead EDs. The varying parameters to evaluate the performance are the number of simultaneous applications and number of EDs.

Figure 4.14. shows the time of the death of any ED in the network versus the number of simultaneous applications and number of EDs in two illustrating graphs with a comparison between Fuzzy-based Dijkstra’s algorithm used in the route discovery approach in one hand, and on the other hand is the regular Dijkstra’s algorithm usage. The difference between these two approaches is mentioned earlier in the Fuzzy and Dijkstra’s algorithms section.

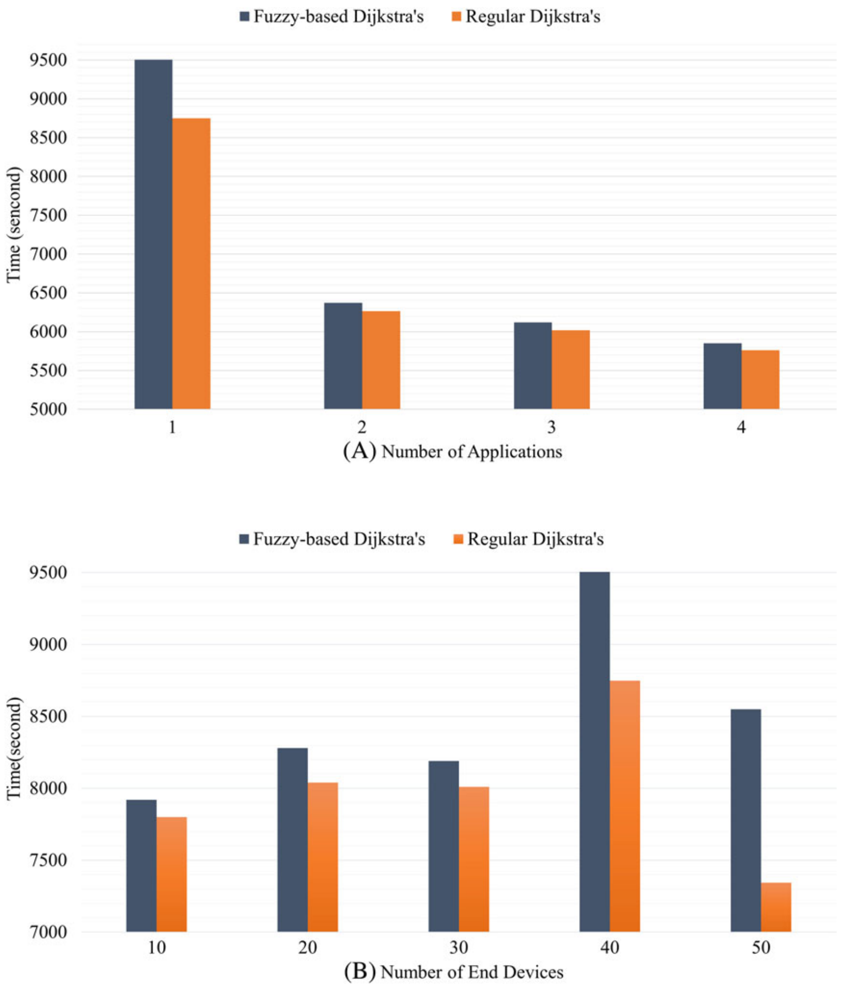


Figure 4.14. The time when the first ED dies for demo 2

It can be observed that there is a decrease in the time when first ED dies, which means more power is consumed that makes the EDs die fast. This is expected because the more applications are granted by SDNC the more power resources of the EDs are consumed. Also, the Fuzzy-based Dijkstra’s algorithm usage prolongs the EDs’ life because it involves the battery level parameter to the calculation of the cost value that will affect the Dijkstra’s algorithm output of the route. Looking at the graph of the number of EDs, the more EDs available in the network the more route alternatives available for the route discovery algorithm to choose. And that makes the SDNC choose another route alternative in case of ED failure trigger. It is noticed that for certain number of EDs, the network life is decreased unexpectedly and that is actually justified by the fact that the simulation environment is random in nature, which makes them produce random seed values that may cause the SDNC does not change the route that will make the first ED dies quickly than normal.

Figure 4.15. displays a comparison of the two mentioned approaches in addition to a ZigBee-based one in case of demonstrating a request of one application and the topology of 50 EDs distributed as shown in Figure 4.10. of the topology of demo 2.

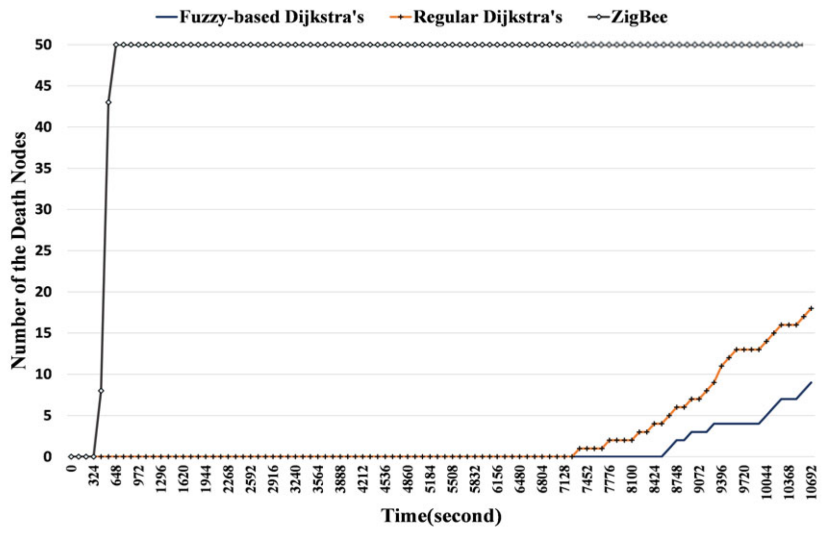


Figure 4.15. Number of dead EDs versus time for demo 2

It’s very obvious how our proposed system using its two approaches outperforms the traditional ZigBee-based platform in terms of the number of EDs die over time. The reason for it is that the ZigBee-based system uses static attributes with some predefined and pre-calculated routes for any application without the ability to change it during duty time as our system can do using both the fuzzy-based Dijkstra’s algorithm and regular Dijkstra’s algorithm. In addition to the uneven distribution of loads over the available resources of the network.

4.7. Conclusion Regarding Simulation Results

A software development of the proposed model was presented in this chapter, preceded by some descriptions of the proposed platform units and modules. The key parts of the proposed platform are topology discovery mechanism; including a dynamic clustering of the network, WSANFlow interface protocol; which is how the SDNC and EDs, communicate, and a route discovery approach; that utilize a Fuzzy-based Dijkstra's algorithm.

Two demonstration simulations were introduced; a preliminary one (demo 1) and an advanced one (demo 2). Demo 1 demonstrated a total of 5 devices, a basic version of interface protocol (WSANFlow) and single application at a time. Demo 2 demonstrated up to 50 devices, a sophisticated version of WSANFlow protocol and multi-application served simultaneously.

The system tested with a variety of traffics categorized as light, and heavier to imply a multimedia stream (details in earlier sections and simulation parameters tables). The results in both demos showed an obvious improvement over the traditional counterpart in terms of power consumption, and an acceptable enhancement in terms of delay and throughput.

CHAPTER 5. PHYSICAL IMPLEMENTATION OF THE PROPOSED ARCHITECTURE

5.1. Introduction

This chapter presents an implementation testbed that was designed and built based on SDN paradigm using equipment and software modules originally created for sensor networks based on classical standardized protocols. We used Texas Instruments devices of system-on-chip CC2538 wireless MCU placed on CC2538EM evaluation module with a development kit of smartrft06 evaluation board that is used for hardware prototyping. an XDS100v3 debug probe is used for software debugging. Code Composer Studio and IAR Embedded Workbench were used as the development environment to develop and debug protocol codes running on the mentioned devices.

Texas Instruments TI provide some protocol suites compatible with low power MCUs like CC2538. Z-stack and TIMAC are one of the protocol suites that are provided by Texas Instruments TI, which can be uploaded to the device to working accordingly. Both of the protocol suites are supported by IEEE 802.15.4 as data link/physical layer standard protocol.

5.2. TI ZigBee Protocol Stack

Z-stack protocol is a protocol compliant suite provided by Texas Instruments for wireless MCUs specialized for low power network devices to develop various sensor networks applications for devices like CC2538. It is based on the popular traditional ZigBee protocol which was designed for low data rate low energy consumption networks. The stack includes ZigBee base device behavior for certain procedures like serving applications, network searching, and network establishment.

The protocol stack can be depicted in Figure 5.1. ZigBee protocol is designed to execute network layer tasks and functions that are placed on top of IEEE 802.15.4 MAC layer standard protocol. Thus Z-stack from TI is a full stack protocol from the application layer to ZigBee based network layer to IEEE 802.15.4 based MAC/PHY layer.

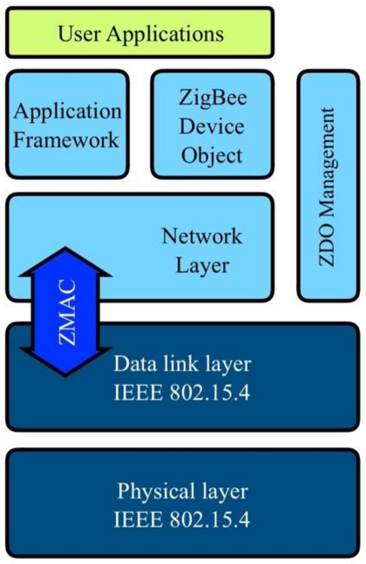


Figure 5.1. Protocol architecture for TI Z-stack

ZigBee stack protocol is developed and demonstrated using IAR Embedded Workbench, which is an IDE with C/C++ build tools in addition to debugger functionalities. Z-stack includes two processors ZAP (Zigbee Application Processor) and ZNP (Zigbee Network Processor). In our case, SmartRF06EB is used as the ZAP that contains different peripherals to be used to demonstrate applications. The ZAP utilizes SoC-based ZNP like CC2538 (used in our research work) to communicate via a ZigBee network [57].

Figure 5.2. depicts how ZAP interacts with SoC based ZNP to connect to a network that is a ZigBee based. The ZAP which runs an application code that uses ZNP API via UART/SPI interface functions to interact with, for example, CC2538-ZNP that runs a full Z-stack and thus connect the application processor to the ZigBee and IEEE 802.15.4 radio network.

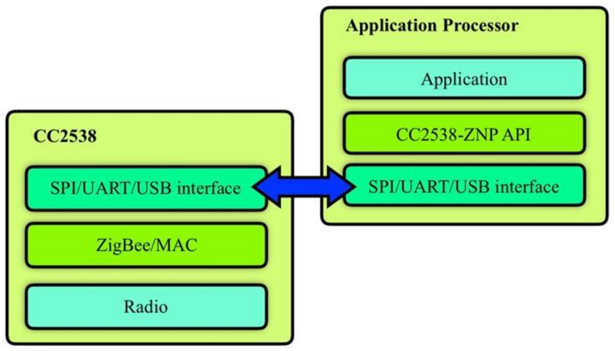


Figure 5.2. ZAP and ZNP interaction

5.3. TIMAC Protocol Stack

TIMAC protocol is an implementation of IEEE 802.15.4 Medium Access Control software in TI wireless MCU devices. That's why it is certified as an amenable protocol to IEEE 802.15.4 standard. TIMAC is included partially in the Z-stack protocol mentioned earlier. It only supports star topology, as it has very plain experimental applications provided by TI.

TIMAC has no defined network functions in its stack except some simple functions alongside some application layer functions in the given stack demo. So, an application for SDN based network-application cross-layer platform is seen as an appropriate choice to be developed on top of the TIMAC module.

The interface between the application layer and lower layers is achieved by OSAL that is an Operating System Abstraction Layer. In the level of the application layer, two main parts of the software are defining the operation of the program flow: initialization and event handling/processing. The initialization part invokes the initiation of different parts of the stack according to the application. The event handling/processing part is defined by the developer/user, which its order must be identical to the task initialization calls. Events may include MAC interrupt, network interrupt, HAL interrupt, certain application interrupt, and special procedures interrupts like green power or fragmentation. HAL is the Hardware abstract layer which is responsible for interface drivers to access services like timers UART, ADC, LCD, LED, KEYS, and I2C services. Figure 5.3. illustrates the layers of a basic TIMAC protocol stack.

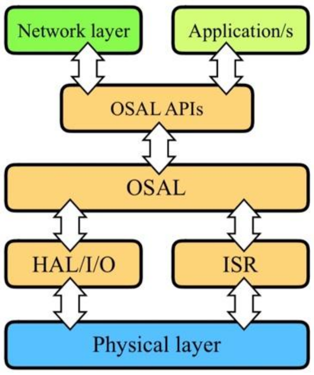


Figure 5.3. The protocol stack of a basic TIMAC

5.4. Physical Environment

A practical testbed of our proposed framework is made by the use of a number of TI devices. The core MCU is TI CC2538, which is ARM Cortex-M3 based powerful SoC. A development kit of smartRF06EB is used to prepare an interface environment for the testbed prototyping.

The TI CC2538 is mounted on an independent chip with an antenna as shown in Figure 5.4., while this chip can be attached to the smartRF06EB to use its peripherals as shown in Figure 5.5.

Figure 5.4. TI CC2538 evaluation module

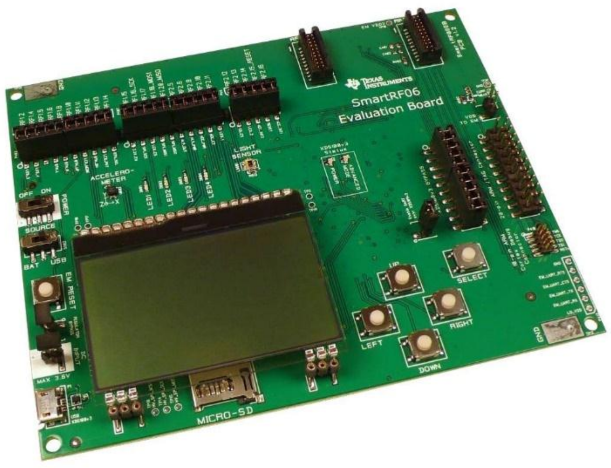


Figure 5.5. SmartRF06EB development kit

SmartRF06EB is equipped with a number of interfaces and features to get a proper environment for various applications and communications abilities, in addition to some debugging features. We may mention a number of features here:

1. TI XDS100v3 emulator for programming and debugging with a capability of adding faster more advanced emulators like TI XDS200v3 and others.

2. High speed USP 2.0 interface that is an integrated serial port for the capability of communicating with other devices like PC through UART channel.

3. LCD and LEDs for demonstrations and debugging as output ports, beside some breakout pins as GPIO input/output ports

4. Accelerometer, light sensor, and buttons as input ports to use for demonstrations and debugging.

5. Micro SD card slot for extra storage and over-the-air programming and more.

6. Various power source options like 1.5 V AAA Alkaline batteries, CR2032 coin cell lithium batteries, USB port power and external independent pins for external power supply.

In our testbed we used two types of physical nodes: one with a full smartRF06EB board with the CC2538EM chip mounted on it as shown in Figure 5.5, and an independent chip of CC2538EM with an independent power source. We used the nodes equipped with smartRF06EB for debugging and providing an application for the demo. The nodes without smartRF06EB (independent CC2538EM chips) are used as extra nodes for forwarding traffics and extending the network.

5.5. Software Environment

To interact, program, debug and even determine the tasks and functions of the testbed with the earlier mentioned devices as a human user, a software tool is needed. These devices have been developed and designed using IAR Embedded Workbench IDE that incorporates compiler, linker, text editor, librarian, assembler, project manager and debugger, which makes it a proper tool for developing protocols and systems

There have been other options to develop such systems for sensor networks in hardware environments such as Contiki which is an open source OS with libraries to support a number of low-power devices for wireless networks, alongside protocols like 6loWPAN, IPv6, and RPL. It has been known for its large active community.

In our research work, we used the commercial IAR Embedded Workbench environment for its robust set of debugging options and ready to use libraries, over the non-commercial Contiki choice for its lack of proper debugging tools and a shortage of need options.

5.6. SDN Layer On Top Of TIMAC

Our choice of protocol to put our developed protocol on top of it, was the TIMAC IEEE 802.15.4 Medium Access Control (MAC) software stack. The reason for this choice:

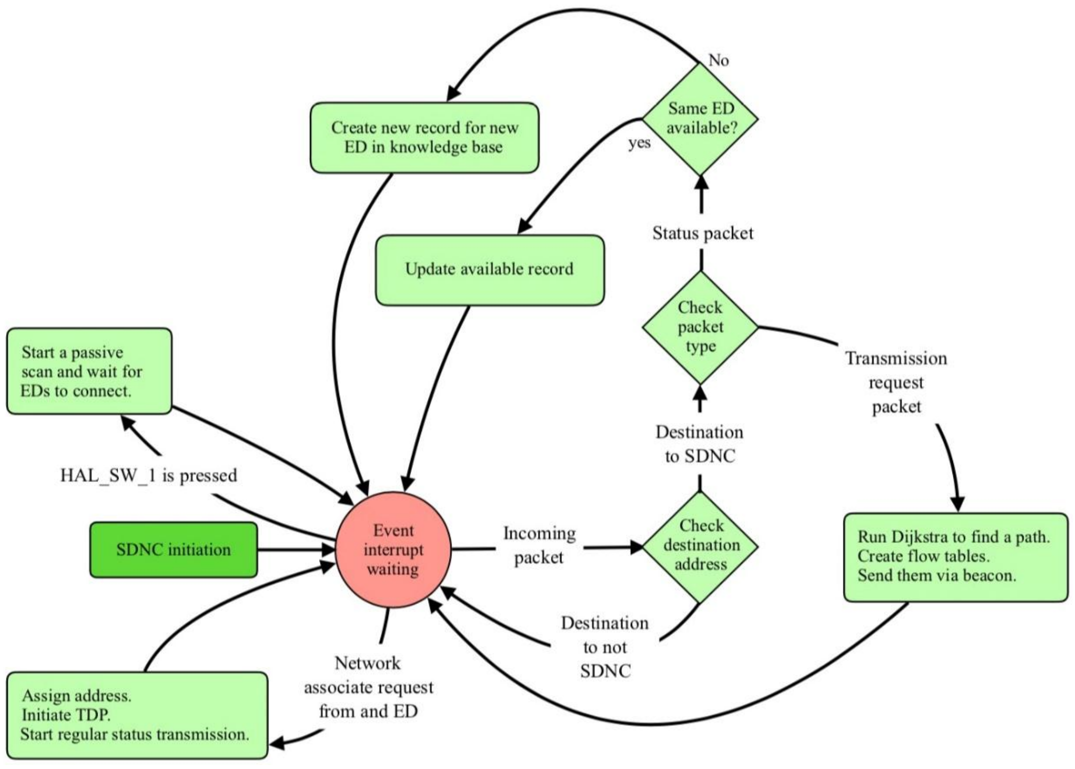
1. Its available library that supports low power sensor network devices MAC protocols.

2. Its available simple application and network protocol testbeds that make it appropriate for customization and development of new protocols for these layers.

3. Its online community of documentation and help desk provided by TI forums.

For these reasons and more, TIMAC was chosen to develop our proposed SDN-based WSAN protocol set utilizing TI CC2538 ARM-based devices to create a testbed to demonstrate and evaluate the developed protocols.

Our proposed SDN based protocol added a specialized layer to the available TIMAC stack. Figure 5.6. shows the event handling state diagram of an SDNC specifically in the layer between the application layer and MAC layer.



No
Same ED available?
yes
Create new record for new ED in knowledge base
Update available record
Status packet
Check packet type
Transmission request packet
Destination to SDNC
Start a passive scan and wait for EDs to connect.
HAL_SW_1 is pressed
SDNC initiation
Event interrupt waiting
Incoming packet
Check destination address
Run Dijkstra to find a path. Create flow tables. Send them via beacon.
Destination to not SDNC
Network associate request from and ED
Assign address. Initiate TDP. Start regular status transmission.

Figure 5.6. The state diagram for event handling in SDNC

Also, an event handling state diagram is shown in Figure 5.7. for an SDN enabled ED to interact with the SDNC.

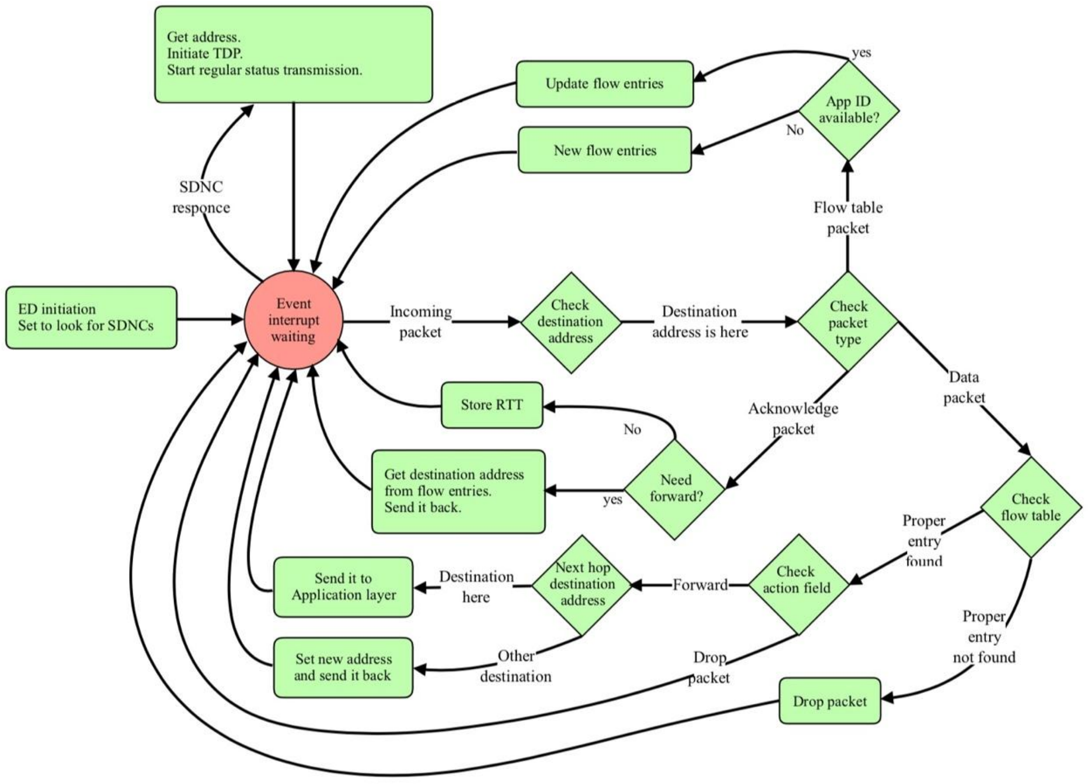


Figure 5.7. State diagram for event handling in ED

5.7. Experimental Layout

We consider a layout of five nodes including the SDNC. The layout looks like what is shown in Figure 5.8., where the deployed devices are shown in a typical scenario. We used smartRF06EB board nodes with the CC2538EM module attached to it, and the CC2538EM board without any peripherals working just as a forwarder.

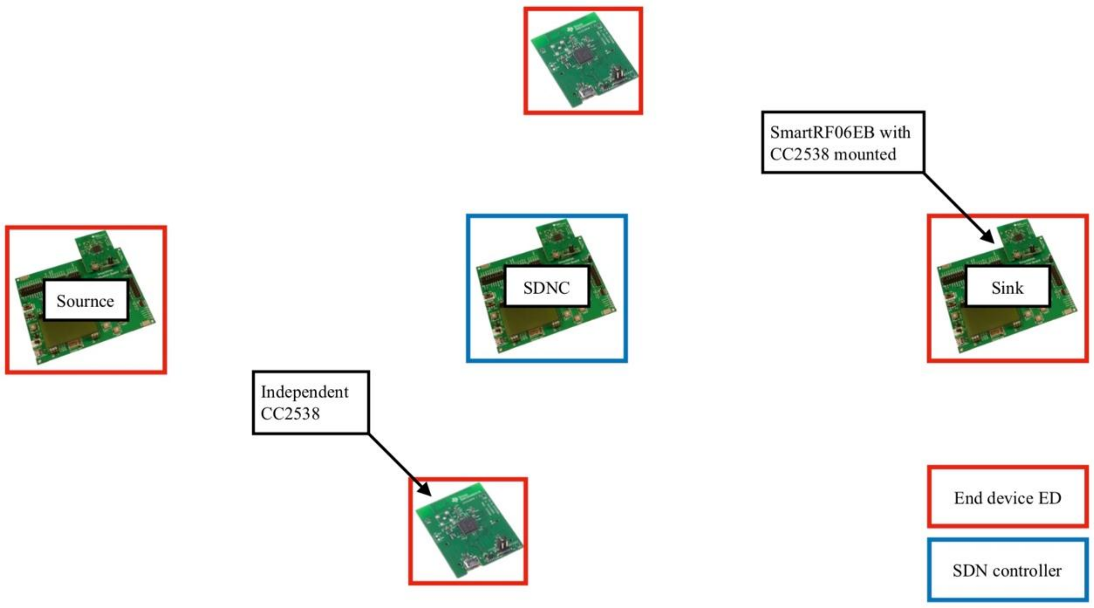


Figure 5.8. The experimented layout for a typical scenario

5.7.1. Environmental conditions and data traffic

For demonstration purposes, we used the built-in smartRF06EB data producing sensors to test our proposed protocol. We used the light sensor and accelerometer sensor, fetch their values then encapsulate them in a packet that will be transmitted through the network with a period of 2 seconds between each consecutive fetched value.

Also, we used the provided buttons in the smartRF06EB board in the SDNC node and the ED nodes as shown in Figure 5.9.

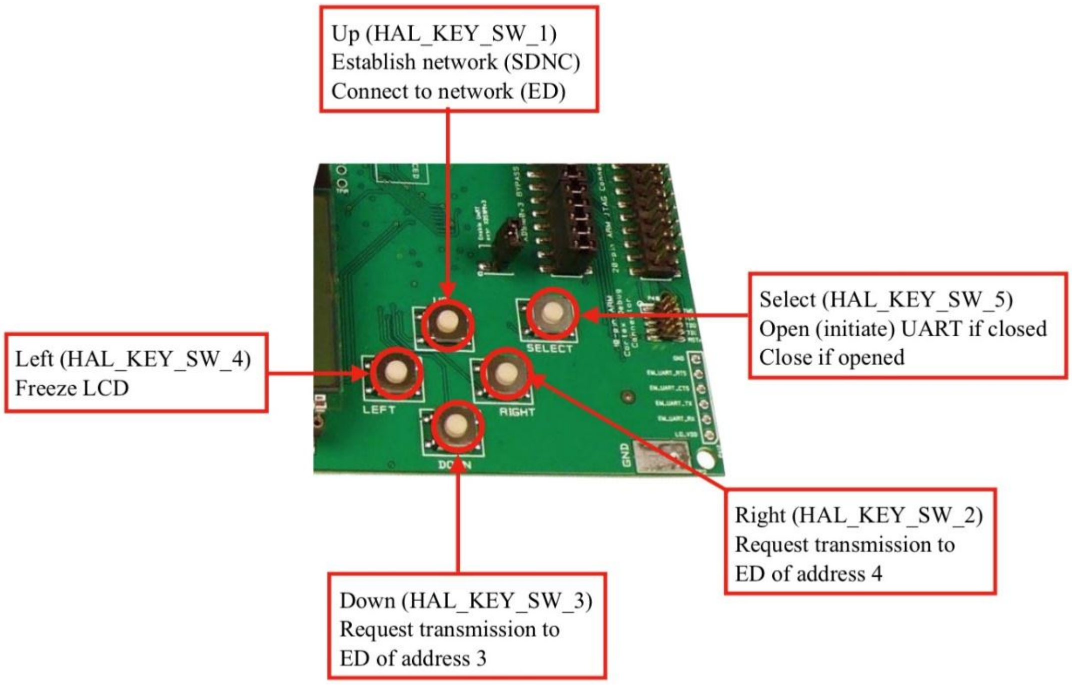


Figure 5.9. SmartRF06EB buttons location and usage in the demonstration

The LEDs in the board were used in the main program to debug and check, for example when the node connects to SDNC network and when there is a received packet, in addition to other uses for debugging and demonstration purposes.

One of the uses of the USB port provided in the smartRF06EB board is the UART communication channel with serial ports of a PC. We used this feature to send specific values from the main program in CC2538 to the PC that the board is connected to. We wrote a Node.js code that was executed in PC to listen to a serial port, get values when available, and display them in a specialized view within a webpage which will be shown in the performance evaluation section of this chapter.

5.7.2. Parameter setting

Several parameters need to be set before the main program is executed. Some of them concern the TIMAC stack for the MAC layer and others concern the application and network layer that have been customized extensively for our proposed protocol.

We set the BO and SO parameters to 1 and they were not changed in duty time. The SDNC node where made to establish the network and start accepting EDs connections after pressing the up button (HAL_KEY_SW_1), while the ED will start looking for SDNCs after 3 seconds of turning on with a flag of joinWithoutButtons is true.

We set maximum devices to connect to SDNC to 20, the maximum number of devices in the neighbor list for a single ED to 10, flow entries for a single ED to 8, and flow entries for SDNC to 32.

Other parameters of MAC and physical layer are set to defaults of IEEE 802.15.4 standard protocol.

5.7.3. Performance assessment and evaluation metrics

In order to debug and examine different aspects of the demonstrated system, we used different tools. Most of them are provided by the smartRF06EB board such as:

1. LEDs: four LEDs that can blink, turn on and off using specific instructions in the program, with other LEDs for power on indication and program upload indication.

2. LCD: a 128 by 64-pixel LCD display that provides 8 lines of 8 pixels high to be used to print integers, strings, and floats. Also, draw lines and specific pixels. The LCD has been used extensively to monitor what is happening inside SDNC and ED.

3. UART connection that is provided in the USB port within the board: this feature has been used to send certain values from the program executed in the devices to the serial port of the PC that the device is connected to using the USB port.

For example, LCD has been used to print incoming status packet content in the SDNC of neighbor list and battery level as shown in Figure 5.10.

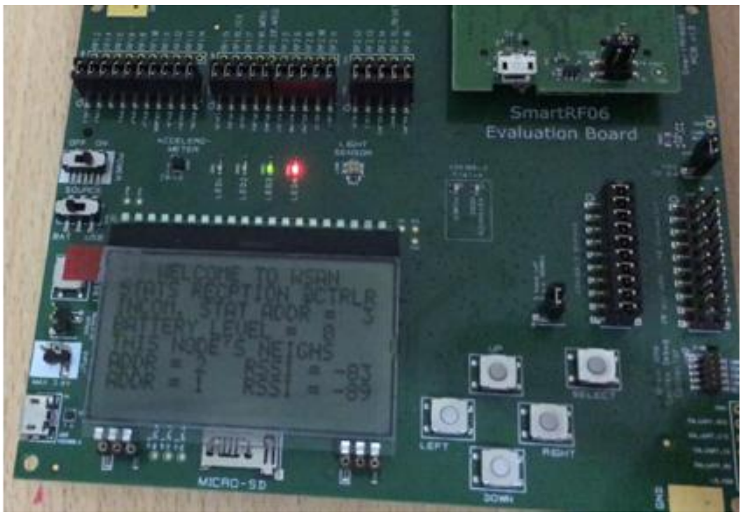
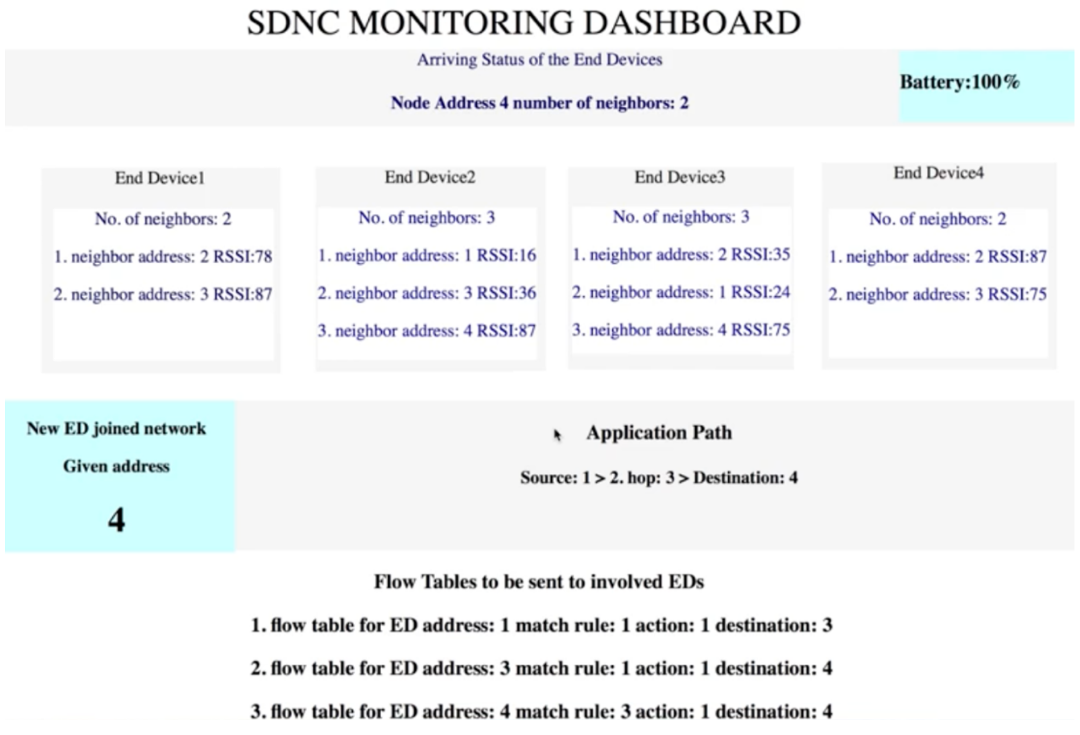


Figure 5.10. LCD displays printed content of a status packet received in SDNC

Another example is the UART connection usage in the demonstration, as it has been used to display values of sensor data of light and accelerometer sensors of a remote source node. Then those values are transmitted through the network to be received at the sink node that is connected to a PC via USB port to get and display the received sensor values. An example of real-time acquirement of these values is shown in Figure 5.11. that shows a picture of the webpage that its varying values have been fetched and injected to HTML page using Node.js codes.



SDNC MONITORING DASHBOARD

Arriving Status of the End Devices

Node Address 4 number of neighbors: 2

Battery:100%

End Device1	End Device2	End Device3	End Device4
No. of neighbors: 2	No. of neighbors: 3	No. of neighbors: 3	No. of neighbors: 2
1. neighbor address: 2 RSSI:78	1. neighbor address: 1 RSSI:16	1. neighbor address: 2 RSSI:35	1. neighbor address: 2 RSSI:87
2. neighbor address: 3 RSSI:87	2. neighbor address: 3 RSSI:36	2. neighbor address: 1 RSSI:24	2. neighbor address: 3 RSSI:75
	3. neighbor address: 4 RSSI:87	3. neighbor address: 4 RSSI:75	

New ED joined network

Given address

4

Application Path

Source: 1 > 2. hop: 3 > Destination: 4

Flow Tables to be sent to involved EDs

1. flow table for ED address: 1 match rule: 1 action: 1 destination: 3

2. flow table for ED address: 3 match rule: 1 action: 1 destination: 4

3. flow table for ED address: 4 match rule: 3 action: 1 destination: 4

Figure 5.11. A picture shows a webpage of received sensor values via UART connection

Two metrics were used to assess the proposed framework performance in the implemented testbed, end to end delay EED, which gives an idea about timings in the system from source to destination. The other metric is the current drawn by a single node, which is an indication of how the power is consumed using the proposed protocol. EED were collected using values sent via UART to the USB port of a PC where it was stored for display. The drawn current is measured using an external apparatus of multimeter attached to a specialized pin in the smartRF06EB board that allow the measurement of the current drawn from the power source by the CC2538EM module mounted on the board.

5.8. Results And Discussion

In this section, we show two kinds of performance measurement criteria which are the end to end delay EED and the drawn current. The EED was measured by adding a procedure of application-to-application acknowledgment and calculate the time of a packet leaving the source node application layer, received at sink node application layer, then coming back to the same source node application layer, divide that by 2 to get the time needed for a packet to travel from source to sink. The EED values were displayed on PC after being received from USB port that is connected to the source device via UART connection. Figure 5.12. shows the results collected for a period of 200 seconds with 7 bytes payload + 7 bytes of packet's header encapsulated with headers from lower layers, and inter-arrival time of 2 seconds.

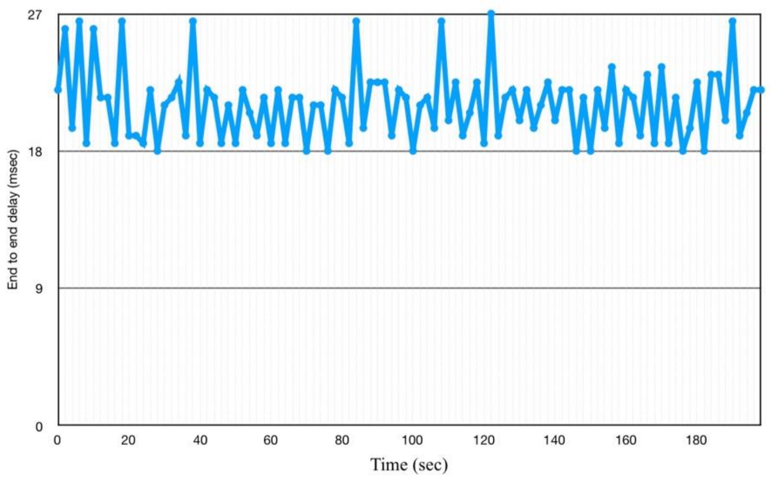


Figure 5.12. EED results

As a physical environment, it can be seen from the figure, that end to end delay results are ranging from 18 to 27 milliseconds with several spikes over the tested period. The demonstration was carried out in a way so that a multi-hop route is chosen and data transmission is realized through it.

The drawn current results are collected using a current measurement header, J503, to measure the current consumption of the CC2538 evaluation module EVM mounted on smartRF06EB board. The header tips are attached by a jumper by default in the normal case. If we put a multimeter probes' heads, we get a current measurement. This measurement should be considered as the current consumed by the EVM on the board. Figure 5.13. shows our tested topology in the situation where we are measuring the drawn current during the tested typical scenario.

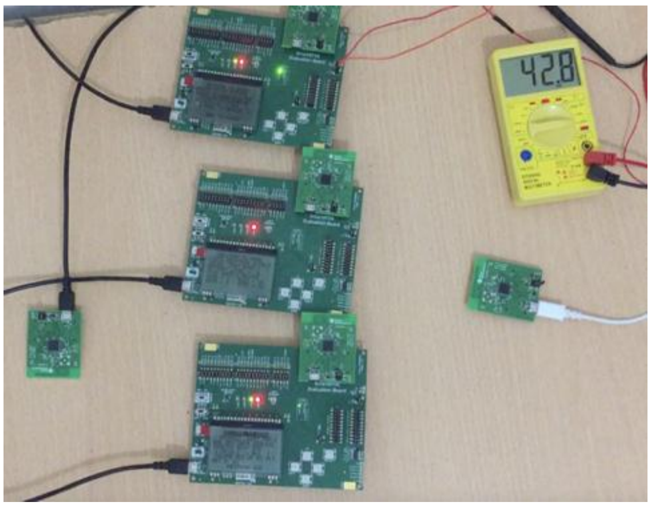


Figure 5.13. Measuring drawn current

Values of the current shown in multimeter are captured and recorded second by second. A 100 second period of time was recorded in terms of drawn current as it can be observed in Figure 5.14.

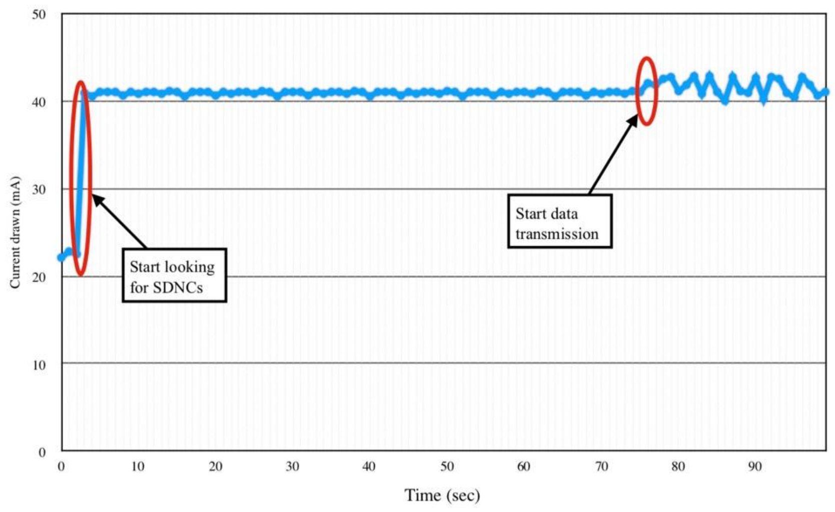


Figure 5.14. Current consumption results

In our research project's website:
http://www.iotlab.sakarya.edu.tr/Projects/Project1.html
accessed September 2018.
There is a video shows the scenario in which we collected the current collection results. The multimeter probes are attached to the sink node. The results start at ~22 mA range of consumption, then after 3 seconds, where an automatic join procedure starts in the device to look for SDNCs in the covered area. At this time (3rd second), the consumption jumps to ~40 mA. When connected to an SDNC the consumption average stays stable, until a transmission request is granted from SDNC and this node starts to receive packets, which requires extra power at nearly the 76th second of node's lifetime.

5.9. Conclusion

An implemented version of our proposed SDN-based WSAN framework was presented in this chapter to observe how SDN-based network functions and modules perform and interact in real devices platform. The proposed protocol modules were placed above the TIMAC protocol that is provided from TI for low current consumption devices. A topology discovery mechanism for neighbor list creation and association was executed. Several procedures were developed and demonstrated like: transmission permission requesting and granting, network-status based route finding, nodes-aware flow table creating and distributing, and in-ED flow-entries based incoming packet forwarding.

The extracted results show a reliable delivery of data packets from source to sink with convenient delay results and considerable power consumption rates. The data conveyed were scaler sensor values like light accelerometer in an indoor lenient environment, but the framework is yet to be tested with more severe data patterns and more acute indoor/outdoor environment.

CHAPTER 6. OVERALL CONCLUSION AND PROBABLE FUTURE TRENDS

We conducted an investigation for an interdisciplinary research if we considered the SDN paradigm existing research trends and sensor networks research trends are two distinct disciplines that working together would unveil a new generation of networks for the fast-paced demands of today's applications. Multiple developments and prototyping environments were employed to design and demonstrate the concept of SDN applied to the core protocols of WSANs for the sake of conducting new behavior and provoke better performances that precedent protocols at the same range of applications. A simulation model was developed and an actual device was prototyped.

Results of both demonstrations showed a promising performance that can be developed in wide scales, as the research in this thesis tackled some significant issues for WSANs such as resource consumption, network complexity, and network configurability. Also, it suggests the centralization of wide-scale, i.e. making SDNC as dominant as it can be for a large number of EDs. Doing that provides both resource management and the usage of as resourceful devices as it can to carry out big data smart algorithms that cannot be put into effect using normal devices.

Probable future trends for our proposed framework might include but not confined:

1. Integration of multiple SDNCs in the same network with the ability to communicate among each other and the usage of shareable resources and information.

2. As mentioned earlier, more advanced algorithms for a huge amount of EDs can be considered in big data scope of processing.

3. Smarter topology discovery can be developed, and more status values can be collected and be involved in routing decision, such as

REFERENCES

[1] A. De Gante, M. Aslan, and A. Matrawy, “Smart wireless sensor network management based on software-defined networking,” *2014 27th Bienn. Symp. Commun.*, pp. 71–75, 2014.

[2] Y. Wang, H. Chen, X. Wu, and L. Shu, “An energy-efficient SDN based sleep scheduling algorithm for WSNs,” *J. Netw. Comput. Appl.*, vol. 59, pp. 39–45, 2016.

[3] W. Xiang, S. Member, N. Wang, and Y. Zhou, “An Energy-Efficient Routing Algorithm for Software-Defined Wireless Sensor Networks,” vol. 16, no. 20, pp. 7393–7400, 2016.

[4] D. A. Babu, “SDN-based WSN Routing Protocol,” 2016.

[5] S. Sharma, R. K. Bansal, and S. Bansal, “Issues and Challenges in Wireless Sensor Networks,” *2013 Int. Conf. Mach. Intell. Res. Adv.*, pp. 58–62, 2013.

[6] K. M. Modieginyane, B. B. Letswamotse, R. Malekian, and A. M. Abu-Mahfouz, “Software defined wireless sensor networks application opportunities for efficient network management: A survey,” *Comput. Electr. Eng.*, vol. 66, pp. 274–287, 2018.

[7] M. Ndiaye, G. Hancke, and A. Abu-Mahfouz, “Software Defined Networking for Improved Wireless Sensor Network Management: A Survey,” *Sensors*, vol. 17, no. 5, p. 1031, 2017.

[8] A. C. G. Anadiotis, S. Milardo, G. Morabito, and S. Palazzo, “Toward Unified Control of Networks of Switches and Sensors Through a Network Operating System,” *IEEE Internet Things J.*, vol. 5, no. 2, pp. 895–904, 2018.

[9] B. R. Al-kaseem and H. S. Al-raweshidy, “SD – NFV as an Energy Efficient Approach for M2M,” *Ieee Internet Things J.*, vol. 4, no. 5, pp. 1–12, 2017.

[10] Y. Duan, W. Li, X. Fu, Y. Luo, and L. Yang, “A methodology for reliability of WSN based on software defined network in adaptive industrial environment,” *IEEE/CAA J. Autom. Sin.*, vol. 5, no. 1, pp. 74–82, 2018.

[11] Q. Yaseen, F. Albalas, Y. Jararwah, and M. Al-Ayyoub, “Leveraging fog computing and software defined systems for selective forwarding attacks detection in mobile wireless sensor networks,” *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 4, pp. 1–13, 2018.

[12] N. Abdolmaleki, M. Ahmadi, H. T. Malazi, and S. Milardo, “Fuzzy topology discovery protocol for SDN-based wireless sensor networks,” *Simul. Model. Pract. Theory*, vol. 79, pp. 54–68, 2017.

[13] B. Van De Velde, “Elaborate Energy Consumption Modelling for OpenWSN,” 2017.

[14] L. Wenxing, W. Muqing, and W. Yuewei, “Design of multi-energy-space-based energy-efficient algorithm in novel software-defined wireless sensor networks,” vol. 13, no. 7, 2017.

[15] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, “Software defined wireless networks: Unbridling SDNs,” *Proc. - Eur. Work. Softw. Defin. Networks, EWSDN 2012*, pp. 1–6, 2012.

[16] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, “A software defined networking architecture for the internet-of-things,” *IEEE/IFIP NOMS 2014 - IEEE/IFIP Netw. Oper. Manag. Symp. Manag. a Softw. Defin. World*, 2014.

[17] Y. Jararweh, M. Al-Ayyoub, A. Darabseh, E. Benkhelifa, M. Vouk, and A. Rindos, “SDIoT: a software defined based internet of things framework,” *J. Ambient Intell. Humaniz. Comput.*, vol. 6, no. 4, pp. 453–461, 2015.

[18] B. Trevizan de Oliveira, R. Cerqueira Afonso Alves, and C. Borges Margi Universidade de São Paulo São Paulo, “Software-Defined Wireless Sensor Networks and Internet of Things Standardization Synergism,” pp. 60–65, 2015.

[19] B. A. Klein, “RPL : IPv6 Routing Protocol for Low Power and Lossy Networks,” *Netw. Archit. Serv.*, no. July, pp. 59–66, 2011.

[20] B. Trevizan De Oliveira, L. Batista Gabriel, and C. Borges Margi, “TinySDN: Enabling multiple controllers for software-defined wireless sensor networks,” *IEEE Lat. Am. Trans.*, vol. 13, no. 11, pp. 3690–3696, 2015.

[21] C. Cao, L. Luo, Y. Gao, W. Dong, and C. Chen, “TinySDM: Software Defined Measurement in Wireless Sensor Networks,” *2016 15th ACM/IEEE Int. Conf. Inf. Process. Sens. Networks, IPSN 2016 - Proc.*, 2016.

[22] A. C. G. Anadiotis, G. Morabito, and S. Palazzo, “An SDN-Assisted Framework for Optimal Deployment of MapReduce Functions in WSNs,” *IEEE Trans. Mob. Comput.*, vol. 15, no. 9, pp. 2165–2178, 2016.

[23] L. Hu, J. Wang, E. Song, A. Ksentini, M. A. Hossain, and M. Rawashdeh, “SDN-SPS: Semi-Physical Simulation for Software-Defined Networks,” *IEEE Sens. J.*, vol. 16, no. 20, pp. 7355–7363, 2016.

[24] R. Pradeepa and M. Pushpalatha, “SDN Enabled SPIN Routing Protocol for Wireless Sensor Networks,” *2016 Int. Conf. Wirel. Commun. Signal Process. Netw.*, vol. 10, no. 6, pp. 639–643, 2016.

[25] C. Buratti, A. Stajkic, G. Gardasevic, S. Milardo, M. D. Abrignani, S. Mijovic, G. Morabito, and R. Verdone, “Testing protocols for the internet of things on the EuWIn platform,” *IEEE Internet Things J.*, vol. 3, no. 1, pp. 124–133, 2016.

[26] H. Fotouhi, M. Vahabi, A. Ray, and M. Bj, “SDN-TAP : An SDN-based Traffic Aware Protocol for Wireless Sensor Networks,” 2016.

[27] Y. J. Chen, L. C. Wang, M. C. Chen, P. M. Huang, and P. J. Chung, “SDN-enabled Traffic-aware Load Balancing for M2M Networks,” *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1797–1806, 2018.

[28] J. A. Puente Fernández, L. Javier, G. Villalba, and T.-H. Kim, “Software Defined Networks in Wireless Sensor Architectures,” pp. 1–24, 2018.

[29] R. Masoudi and A. Ghaffari, “Software defined networks: A survey,” *J. Netw. Comput. Appl.*, vol. 67, pp. 1–25, 2016.

[30] R. Friedman and D. Sainz, “An Architecture for SDN Based Sensor Networks,” *Proc. 18th Int. Conf. Distrib. Comput. Netw. - ICDCN ’17*, pp. 1–10, 2017.

[31] A. B. A. C. C, “WSANFlow : An Interface Protocol Between SDN Controller and End Devices for SDN-Oriented WSAN,” 2018.

[32] Y. Wei, X. Ma, N. Yang, and Y. Chen, “Energy-Saving Traffic Scheduling in Hybrid Software Defined Wireless Rechargeable Sensor Networks,” *Sensors*, vol. 17, no. 9, p. 2126, 2017.

[33] O. N. Foundation, “ONF SDN Evolution,” 2016.

[34] P. Goransson, C. Black, and T. Culver, *Software Defined Networks: A Comprehensive Approach - 2nd Ed*. 2016.

[35] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, “A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements,” *IEEE Access*, vol. 5, pp. 1872–1899, 2017.

[36] Open Networking Foundation, “OpenFlow Switch Specification 1.5.1,” 2015.

[37] S. V. Manisekaran and R. Venkatesan, “An analysis of software-defined routing approach for wireless sensor networks,” *Comput. Electr. Eng.*, vol. 56, pp. 456–467, 2016.

[38] Open Networking Foundation, “SDN Architecture Overview,” 2013.

[39] J. Wang, Y. Miao, P. Zhou, M. S. Hossain, and S. M. M. Rahman, “A software defined network routing in wireless multihop network,” *J. Netw. Comput. Appl.*, vol. 85, no. November 2016, pp. 76–83, 2017.

[40] L. Tello-oquendo, I. F. Akyildiz, S. Lin, and V. Pla, “SDN-Based Architecture for Providing Reliable Internet of Things Connectivity in 5G Systems,” *2018 17th Annu. Mediterr. Ad Hoc Netw. Work.*, pp. 1–8.

[41] K. Yap, M. Kobayashi, R. Sherwood, T. Huang, M. Chan, N. Handigol, and N. Mckeown, “OpenRoads : Empowering Research in Mobile Networks,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 125–126, 2010.

[42] L. E. Li, Z. M. Mao, and J. Rexford, “Toward software-defined cellular networks,” *Proc. - Eur. Work. Softw. Defin. Networks, EWSDN 2012*, pp. 7–12, 2012.

[43] S. Zerkane, D. Espes, P. Le Parc, and F. Cuppens, “Vulnerability Analysis of Software Defined Networking,” in *International Symposium on Foundations and Practice of Security Springer*, 2016, pp. 97–116.

[44] Z. Yao and Z. Yan, “A trust management framework for software□defined network applications,” *Concurr. Comput. Pract. Exp.*, no. March, pp. 1–18, 2018.

[45] A. Jasim and C. Ceken, “Video streaming over wireless sensor networks,” *Wirel. Sensors (ICWiSe), 2015 IEEE Conf.*, pp. 63–66, 2015.

[46] Z. Bidai, “Interference-Aware Multipath Routing Protocol for Video Transmission over ZigBee Wireless Sensor Networks,” *Int. Conf. Multimed. Comput. Syst.*, pp. 837–842, 2014.

[47] T. Wiegand, G. J. Sullivan, S. Member, G. Bjøntegaard, A. Luthra, and S. Member, “Overview of the H . 264 / AVC Video Coding Standard,” vol. 13, no. 7, pp. 560–576, 2003.

[48] F. Korbel, *FFmpeg Basics: Multimedia handling with a fast audio and video encoder by Frantisek Korbel*, 1st ed. CreateSpace Independent Publishing, 2012.

[49] A. Klein and J. Klaue, “Performance Evaluation Framework for Video Applications in Mobile Networks,” *2009 Second Int. Conf. Adv. Mesh Networks*, pp. 43–49, 2009.

[50] F. H. P. Fitzek and M. Reisslein, “MPEG-4 and H.263 video traces for network performance evaluation,” *IEEE Netw.*, vol. 15, no. 6, pp. 40–54, 2001.

[51] F. Van Der Schueren, J. Doggen, and V. Der Schueren, “Design and Simulation of a H.264 AVC Video Streaming Model,” *Proc. Third Eur. Conf. Use Mod. Inf. Commun. Technol. ECUMICT 2008*, 2008.

[52] “OPEN ZigBee Poject.” [Online]. Available: www.open-zb.net.

[53] M. Krunz and H. Hughes, “‘A Traffic Model for MPEG-Coded VBR Streams,’” *Proc. SIGMETRICS*, pp. 47–55, 1995.

[54] S. I. Society, “IEEE Standard for Local and metropolitan area networks — Part 15 . 4: Low-Rate Wireless Personal Area Networks (LR-WPANs),” 2011.

[55] K. Sood, S. Yu, and Y. Xiang, “Software-Defined Wireless Networking Opportunities and Challenges for Internet-of-Things: A Review,” *IEEE Internet Things J.*, vol. 3, no. 4, pp. 453–463, 2016.

[56] M. Al-hubaishi, C. Çeken, and A. Al-shaikhli, “A novel energy□aware routing mechanism for SDN□enabled WSAN,” *Int J Commun Syst.*, 2018.

[57] T. Instruments, “Z-Stack User ’ s Guide For CC2530 ZigBee-PRO Network Processor Sample Applications,” 2010.

RESUME

Ali Burhan Alshaikhli was born 1984 in Baghdad. He received his B.S. and M.S. degrees in computer engineering from Nahrain University, Iraq, in 2006 and 2010, respectively. He is currently pursuing his Ph.D. degree in computer and information engineering at Sakarya University. He is expected to be graduated in 2018.