



**REPUBLIC OF TURKEY
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND
TECHNOLOGY UNIVERSITY**

**GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCES
DEPARTMENT OF NANOTECHNOLOGY AND ENGINEERING
SCIENCES**

**TRANSMITTING VITAL HEALTH DATA WITH STANDART
INTERNET OF THINGS PROTOCOLS**

**SEDAT BİLGİLİ
MASTER OF SCIENCE**



REPUBLIC OF TURKEY

ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF NANOTECHNOLOGY AND ENGINEERING SCIENCES

**TRANSMITTING VITAL HEALTH DATA WITH STANDART INTERNET OF
THINGS PROTOCOLS**

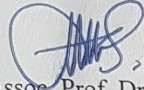
SEDAT BİLGİLİ

MSc THESIS

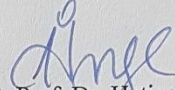
ADANA 2019

TRANSMITTING VITAL HEALTH DATA WITH STANDART INTERNET OF THINGS PROTOCOLS

Submitted by **Sedat BİLGİLİ** in partial fulfillment of the requirements for the degree of **Master of Science in Department of Nanotechnology and Engineering Sciences, Adana Alparslan Türkeş Science and Technology University** by,

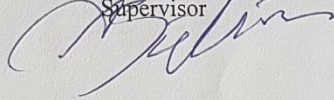


Assoc. Prof. Dr. Osman
SIVRIKAYA
Graduate School Director



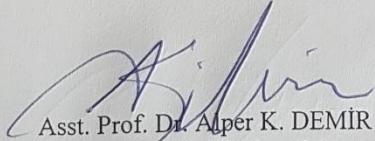
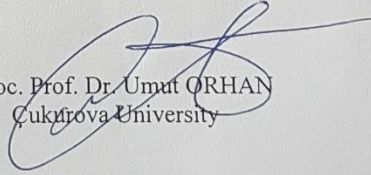
Asst. Prof. Dr. Hatice İmge
OKTAY BAŞEĞMEZ
Department Chair

Asst. Prof. Dr. Alper K. DEMİR
Supervisor

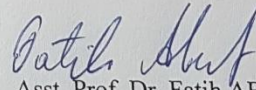


Thesis Committee

Assoc. Prof. Dr. Umut ORHAN
Çukurova University



Asst. Prof. Dr. Alper K. DEMİR
Adana Alparslan Türkeş Science
and Technology University



Asst. Prof. Dr. Fatih ABUT
Adana Alparslan Türkeş Science
and Technology University

Thesis Defense Date

24 / 05 / 2019

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


Sedat BILGILI

TRANSMITTING VITAL HEALTH DATA WITH STANDART INTERNET OF THINGS PROTOCOLS

BİLGİLİ Sedat

Master of Science, Nanotechnology and Engineering Sciences

May 2019, 58 Pages

ABSTRACT

The Internet of Things (IoT) is one of the most popular technologies of today. In recent years, the use of the IoT in daily life is being increased and it is foreseen to increase further over time. In addition to being able to communicate in an autonomous way without the need for any user, IoT devices can obtain data via the sensors on them, or they can process the incoming data. The IoT concept can be used in factory automation, intelligent traffic management systems, smart city systems, home automation systems and many other areas. The health sector is also one of the areas where IoT systems are used. With IoT systems, health data of patients can be acquired autonomously remotely. Patients under follow-up may choose to remain at home or in nursing homes due to unfavorable hospital conditions, crowds and cost reasons. As a solution to these situations, IoT systems can be considered to collect health data and transmit them to a health center. There are different network protocol stacks that can work with IoT systems. In the case of transmission of the health data in question, it is important that this protocol stack can carry health data in the most accurate and fastest way. In this study, the usability of these protocol stacks in health data transmission was examined and the performances of the available protocol stacks were analyzed.

Keywords: *Internet of Things, Wireless Sensor Networks, 6LoWPAN, Medium Access Control, Radio Duty Cycle, CoAP*

STANDART NESNELERİN İNTERNETİ PROTOKOLLERİ İLE HAYATİ SAĞLIK VERİLERİNİN TAŞINMASI

BİLGİLİ Sedat

Yüksek Lisans, Nanoteknoloji ve Mühendislik Bilimleri

Mayıs 2019, 58 Sayfa

ÖZET

Nesnelerin interneti, günümüzde oldukça popüler olan teknolojilerden birisidir. Son yıllarda, nesnelerin internetinin günlük hayattaki kullanımları artmaktadır ve zaman içerisinde daha da artması öngörülmektedir. IoT cihazları, herhangi bir kullanıcıya gerek kalmaksızın, otonom şekilde iletişim yapabilmelerinin yanı sıra, üzerlerinde bulunan sensörler aracılığı ile veri elde edebilir, veya kendilerine gelen verileri işleyebilirler. Nesnelerin interneti konsepti, fabrika otomasyonları, akıllı trafik yönetim sistemleri, akıllı şehir sistemleri, ev otomasyon sistemleri ve daha birçok alanda kullanılabilir. Sağlık sektörü de IoT sistemlerinin kullanıldığı alanlardan birisidir. IoT sistemleri ile, hastaların sağlık verileri otonom olarak uzaktan elde edilebilir. Takip altında olması gereken hastalar, elverişsiz hastane koşulları, kalabalık ve maliyet gibi olumsuz nedenler yüzünden evde veya bakım evlerinde kalmayı tercih edebilir. Bu durumlara çözüm olarak, sağlık verilerini toplayarak bir sağlık merkezine iletebilen IoT sistemleri düşünülebilir. IoT sistemleri ile çalışabilen farklı ağ protokol yığınları mevcuttur. Söz konusu olan sağlık verilerinin iletimi olduğunda, bu protokol yığınının, sağlık verilerini en doğru ve en hızlı şekilde taşıyabilmesi önemlidir. Bu çalışma kapsamında, bu protokol yığınının, sağlık verileri iletiminde kullanılabilirliği incelenmiştir ve kullanılabilir olan yığınların ise performansları analiz edilmiştir.

Anahtar Kelimeler: *Nesnelerin İnterneti, Kablosuz Algılayıcı Ağları, Ortama Erişim Kontrolü, Radyo Görev Döngüsü, 6LoWPAN, CoAP*

ACKNOWLEDGEMENTS

First of all, I would like to thank Asst. Prof. Dr. Alper K. DEMİR for his supports and efforts throughout my work. Without his support and guidance, I could never complete this study.

I would like to thank each and every member of the evaluation committee for their guidance.

I would like to thank my wife, Ebru BİLGİLİ, who was with me during the study with her support and patience. She always kept my motivation alive by being with me all the time.

I would like to thank my family, my father Sami BİLGİLİ, my mother L. Vahide BİLGİLİ and my sister Seda BİLGİLİ for their support.

I would also like to thank my friends Hüseyin Emre ÖZGÜR and Halit DEMİR for their support in this study.

This thesis was financially supported by SIREN project funded by the Scientific and Technological Research Council of Turkey (TUBİTAK) under Grant No 116E025.

TABLE OF CONTENTS

ABSTRACT	iii
ÖZET	iv
ACKNOWLEDGEMENTS.....	v
TABLE OF CONTENTS	vi
LIST OF FIGURES	viii
LIST OF TABLES	x
NOMENCLATURE	xi
1. INTRODUCTION.....	1
1.1. Internet of Things	1
1.2. Suitable IoT Environment for Collecting Health Data	1
1.3. Health Sensors & IoT Environment.....	2
2. PROBLEM STATEMENT	3
3. LITERATURE REVIEW	4
4. MATERIALS AND METHODS.....	7
4.1. Materials.....	7
4.1.1. Software environment.....	7
4.1.1.1. Cooja network simulator	7
4.1.1.2. ContikiOS	8
4.1.1.3. WisMote (Used in simulation)	9
4.1.1.4. Californium.....	10
4.1.2. Hardware Environment	10
4.1.2.1. IoT device: OpenMote.....	11
4.1.2.2. Health sensors and other equipments	12
4.1.3. Network Stack.....	13
4.1.3.1. Physical layer: 802.15.4 radio	13
4.1.3.2. Radio duty cycling (RDC)	14
4.1.3.3. Medium access control (MAC)	15

4.1.3.4.	Network: 6LoWPAN	16
4.1.3.5.	Network: IPv6	16
4.1.3.6.	Network: RPL routing	17
4.1.3.7.	Transport: UDP	18
4.1.3.8.	Application: CoAP.....	19
4.2.	Methods.....	20
4.2.1.	Obtaining health data.....	20
4.2.2.	Determination of vital health data traffic characteristics.....	22
4.2.3.	Preparation of simulation environment	24
4.2.3.1.	Cooja network tool & contiki os.....	24
4.2.3.2.	Californium coap client	25
4.2.4.	Performance metrics.....	25
4.2.4.1.	Latency	26
4.2.4.2.	Energy efficiency	26
4.2.4.3.	Reliability.....	27
4.2.4.4.	Throughput.....	27
4.2.5.	Network topology	27
5.	RESULTS AND DISCUSSION.....	29
5.1.	Latency.....	29
5.2.	Energy Efficiency	34
5.3.	Reliability	40
5.4.	Throughput	44
5.4.1.	Node Throughput	44
5.4.2.	NetworkThroughput	48
5.5.	Selecting the Valid Protocol Stacks	51
5.6.	Determination of Optimal IoT Stack to Transfer Health Data	52
6.	CONCLUSION.....	54
7.	REFERENCES	56

LIST OF FIGURES

Figure 1.1 - An Example of IoT Environment	1
Figure 3.1 - Cooja Network Simulator.....	8
Figure 3.2 - Zolertia z1 (left) and WisMote (right) IoT nodes.....	10
Figure 3.3 - OpenMote, OpenBattery, OpenBase, OpenUSB, OpenMote Rev.A1 (left to right)	11
Figure 3.4 - MySignals HW Health Sensor Board.....	12
Figure 3.5 - Arduino UNO Microcontroller Board	12
Figure 3.6 - Network Layers of LLN (left) and traditional Internet Stack (right).....	13
Figure 3.7 - 802.15.4 Radio Channels	14
Figure 3.8 - IPv6 Header	17
Figure 3.9 - UDP Header.....	18
Figure 3.10 - CoAP CON (left) and NON (right) Message Types	20
Figure 3.11 - Hardware Setup Used to Collect Health Data.....	21
Figure 3.12 - Network Topology	28
Figure 4.1 - Latency (Maximum Delay Between Two Packets) graphic of nullMAC / nullRDC scenario.....	31
Figure 4.2 - Latency (Maximum Delay Between Two Packets) graphic of CSMA / nullRDC scenario.....	32
Figure 4.3 - Latency (Maximum Delay Between Two Packets) graphic of nullMAC / contikiMAC scenario	33
Figure 4.4 - Latency (Maximum Delay Between Two Packets) graphic of CSMA / contikiMAC scenario	33
Figure 4.5 - Average Energy Consumption per node for PDR=100	36
Figure 4.6 - Average Energy Consumption per node for PDR=95	37
Figure 4.7 - Average Energy Consumption per node for PDR=90	37
Figure 4.8 - Average Energy Consumption per node for All Network Stacks	38
Figure 4.9 - Average Battery Life of Nodes in Different Network Stacks	39
Figure 4.10 – Successful Packet Ratio for PDR = 100.....	42
Figure 4.11 – Successful Packet Ratio for PDR = 95	42
Figure 4.12 – Successful Packet Ratio for PDR = 90	43
Figure 4.13 – Average Successful Packet Ratio for Different Network Stacks.....	43
Figure 4.14 - Node Throughput for PDR = 100.....	45
Figure 4.15 - Node Throughput for PDR = 95.....	46
Figure 4.16 - Node Throughput for PDR = 90.....	47

Figure 4.17 - Average Node Throughput for Different Network Stacks	47
Figure 4.18 - Network Throughput for PDR = 100	49
Figure 4.19 - Network Throughput for PDR = 95	50
Figure 4.20 - Network Throughput for PDR = 90	50
Figure 4.21 - Average Network Throughput for Different Protocol Stacks	51



LIST OF TABLES

Table 3.1 - Health Sensors and Data	23
Table 3.2 - Power Consumption of two different 802.15.4 Radio ICs	26
Table 4.1 - Health Sensor Data Acquisition Interval Threshold Values by Patient Categories	29
Table 4.2 - Maximum Delay Between Packets (in minutes) in different scenarios	30
Table 4.3 - Average Energy Consumption per Node	35
Table 4.4 - Battery Life per Node (in Days)	39
Table 4.5 – Average Successful Packets	41
Table 4.6 - Node Throughput (Packet per Second)	45
Table 4.7 - Network Throughput (Packet per Second).....	48
Table 4.8 - Ideal Protocol Stack for Different Groups (for up to 15 nodes).....	52
Table 4.9 - Ideal Protocol Stack for Different Groups (for 15-30 nodes).....	53

NOMENCLATURE

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Network(s)
ACK	Acknowledgement
BP	Blood Pressure
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
CTS	Clear to Send
CoAP	Constrained Application Protocol
CoCoA	CoAP Simple Congestion Control-Advanced
DAG	Directed Acyclic Graph
DAO	Destination Advertisement Objects
DIO	DODAG Information Object
DIS	DODAG Information Solicitations
DODAG	Destination-Oriented Directed Acyclic Graph
DTLS	Datagram Transport Layer Security
ECG	Electrocardiogram
EEG	Electroencephalogram
EMG	Electromyogram
HW	Hardware
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv4	Internet Protocol v4
IPv6	Internet Protocol v6
IoT	Internet of Things
JTAG	Joint Test Action Group
KB	Kilo Bytes
LLN	Low Power and Lossy Network
mA	Milliampere
MAC	Medium Access Control
NAT	Network Address Translation
OSI	Open Systems Interconnection
PDR	Packet Delivery Ratio
PPS	Packet Per Second
RAM	Random Access Memory
RDC	Radio Duty Cycle
RPL	Routing Protocol for Low Power and Lossy Networks

RTS	Request to Send
RTT	Round Trip Time
RX	Receive / Receiver
SPo2	Oxygen Saturation
TCP	Transmission Control Protocol
TX	Transmit / Transmitter
UDP	User Datagram Protocol
WSN	Wireless Sensor Network(s)



1. INTRODUCTION

1.1. Internet of Things

Internet of Things (Atzori, Iera, and Morabito 2010; Bandyopadhyay and Sen 2011; Kortuem et al. 2010; Li, Da Xu, and Zhao 2015; Mainetti, Patrono, and Vilei 2011), with the simplest definition, can be expressed as the objects that can be used in daily life connected to the internet and exchanging data. The devices that can be used in these environments are low cost, constrained devices. The fact that these devices are connected directly to the internet will bring a cost burden. Usually, the approach used includes nodes that collect sensors and send data to a gateway device. This gateway device is directly connected to the Internet network. Thus, it acts as a bridge between Internet and IoT devices. Data presented to the Internet can be viewed, processed and stored. Such an IoT structure is presented visually in Figure 1.1.

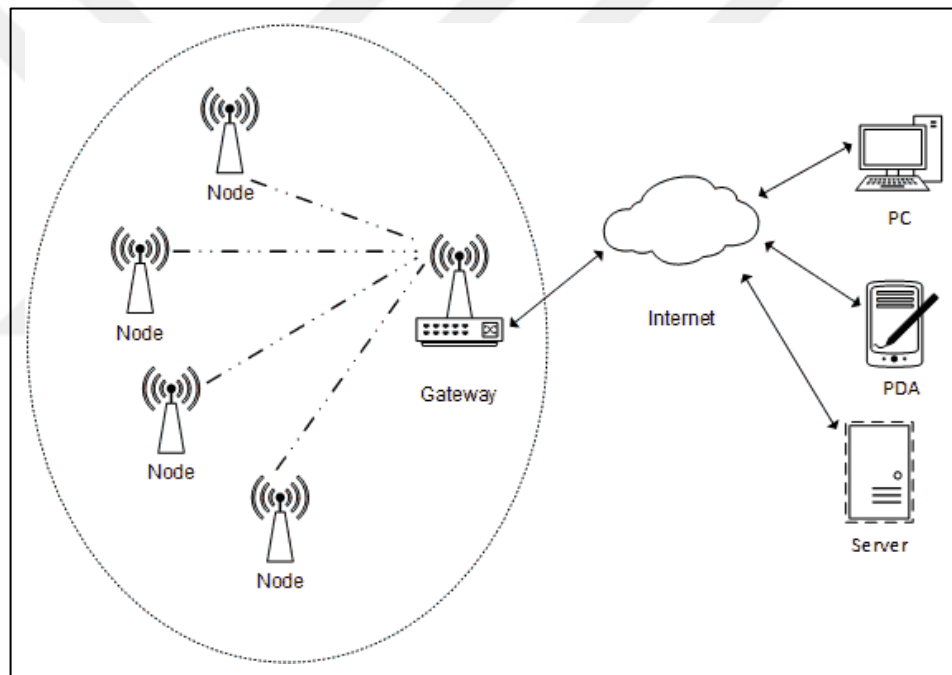


Figure 1.1 - An Example of IoT Environment

1.2. Suitable IoT Environment for Collecting Health Data

The collection of health data from individuals is one of the subjects that have been studied in the literature for a long time. In most studies, sensors connected to individuals transmit data via cables. In more recent studies, health data have been transmitted in a wireless environment. Different technologies were used to transfer health data wirelessly. The different technologies used have brought different problems to be solved. Bluetooth, which is one of the technologies that can be used to transmit health data in wireless environment, is very limited in terms of communication range. When Wi-Fi technology is

preferred with a different approach, high energy consumption is concerned. For these reasons, low power consumption technologies with high communication range have been investigated. Among the wireless options, the most noticeable is the 802.15.4 radio. In addition to its low power consumption, the 802.15.4 radio has a higher communication range than its alternatives.

1.3. Health Sensors & IoT Environment

Recent advances in electronics and integrated circuits has pioneered the development of wearable or implantable little and smart medical devices on human body. Through these devices, physiological data on human body can be gathered. Once these devices are equipped with wireless communication unit, they realize data gathering and communication features in a network environment. For example, with a pulse oximeter, heartbeat, oxygen saturation level in the blood and change of blood volume level on the skin of people can be measured. With an ECG sensor, heart functions of people can be tracked. With an EMG sensor, muscle functions of people can be tracked. With an EEG sensor, brain activities of people can be tracked. With a BP sensor, organ damage and cardiovascular movements of people can be tracked. With an accelerometer sensor, activities of people can be tracked. Such a wireless communication network can be used, for example, in a dispensary (home for elderly people who can no longer care for themselves) to keep track of patients. When the physiological data of patients is transferred to a patient monitoring center (for example to a hospital) over the Internet, the health status of the patients can be monitored remotely.

Some of these sensors have been developed in research laboratories or industry, and they can be used in Internet of Things ambient. In this work, the most efficient standard Internet of Things protocol configurations and algorithms transferring vital signs of individuals equipped with health physiological sensors to a patient monitoring center problem has been studied. Namely, vital signs of individuals will be transferred to a gateway over CoAP/UDP/IPv6/6LowPAN/802.15.4 protocol stack using RPL (Gaddour and Koubâa 2012) routing protocol within an Internet of Things environment, and from gateway to a patient monitoring center. CoAP (Bormann, Castellani, and Shelby 2012; Shelby, Hartke, and Bormann 2014), 6LowPAN (Chen et al. 2011), RPL protocols are standardized by IETF, and 802.15.4 protocol is standardized by IEEE for low power and lossy IPv6 networks. These protocols will shape the future Internet of Things applications. In addition to successful application operation metric, energy, delay, reliability and throughput metrics has been used in performance evaluations. In brief, within the scope of this work, transferring vital signs of individuals to a patient monitoring center by the most efficient Internet of Things network protocol configurations and algorithms has been studied.

2. PROBLEM STATEMENT

In the IoT environment, different network layer protocols can be used for communication purposes and different network topologies may occur depending on the circumstances. Selection of wrong network protocol stack may make the network environment more lossy. In order to deliver vital health data in such an environment, it is important to ensure a minimum latency. In a network environment where health data is transported, packet losses can lead to critical results. In order to avoid these critical results, network protocols should be designed so that the traffic is as lossless as possible. Different network layer protocols can be applied in order to create a network with low loss rate and high reliability. There may even be a network layer protocol combination that can provide a low loss rate in all topologies.

In an environment where critical health data is transmitted, besides the lossless transmission of data, another important issue is energy consumption. Health sensors and the IoT device must be located on the person to collect and transmit health data. This situation may be uncomfortable for the person. Therefore, the device(s) to be installed should be minimized as much as possible. Moreover, the person is connected to these devices may be unable to deal with operations such as battery replacement. The reduction of energy consumption can provide a smaller sized battery usage. In addition, low energy consumption will require less battery replacement. For these reasons, when selecting a protocol stack, power consumption is also an important issue to be taken into consideration.

In addition to these criteria, it is necessary to transfer the health data to be transferred to a server in a correct and meaningful way. Packet losses should be minimized as well as packet corruptions should be prevented.

Patients whom health data to be collected, may be in the form of crowded groups. In such environments, the throughput that the IoT network can carry is also considered. Knowing how many patients can receive data from an IoT network is important for a designed health data collection system. Exceeding the expected number of patients can lead to an increase in data traffic volume and, consequently, a network traffic congestion that cannot be avoided.

In summary, in the IoT environment, the protocol stack needed to transport health data in the fastest, most lossless, most energy efficient and most efficient way should be investigated. In the relevant IoT environment, it is not enough to provide these conditions in the best way possible. Health data may have different needs. These requirements should be investigated and the ideal IoT protocol stack should be determined based on these requirements.

3. LITERATURE REVIEW

The transmission and processing of the data received from the health sensors in the IoT environment is a subject that is becoming increasingly widespread and is expected to increase further in the future. This will inevitably lead to improvements and advancements in the field of healthcare (Lo, Ip, and Yang 2016; Martínez-Caro et al. 2018). There are many studies in the literature on the transmission of health data through the Internet of Things. Some of these studies use the 6LoWPAN infrastructure, while others use standard Internet protocol infrastructure with technologies such as Wi-Fi, Bluetooth. Although studies are mostly proof-of-concept, there are also studies that include performance analysis or new network protocols.

One of these studies (Swaroop et al. 2019), compares GSM, Bluetooth Low-Energy and SMS technologies while collecting health sensor data. The scope of the study is based on the comparison of health data collection performance over more standardized protocols. In the study which does not have energy consumption assessment, 6LoWPAN structure is not used. In another study (Irman 2018), a heart rate sensor and a button on the patient were used to collect health data over standard Wi-Fi network. When the patient's pulse goes out of range or when the patient presses the button, the health data is transmitted to the center. The study is more about proving that the system is working correctly. It is inevitable that the concept of IoT is merged with the popular technology of cloud technology. One of the studies combines cloud and IoT concept (Wan et al. 2018). In this study, heart rate data of the patients were collected and transmitted to the cloud environment. There is also another study on integrating IoT networks with cloud systems (Muhammad et al. 2017). The aim of the study is to transfer the data received from IoT devices with limited storage and communication to the cloud environment and to decompose the meaningful data with a layer. In this study, a framework that offers these features was proposed. The ability to monitor, process and manage patient data in a cloud environment and to be able to distinguish the health data obtained by classification methods (Hassanalieragh et al. 2015) may facilitate patient follow-up. While the cloud environment is well suited for such IoT applications, there may be problems such as delays due to heavy traffic. To prevent these problems, an intermediate layer, fog (Tran et al. 2018; Wayangankar and Prakash Jorvekar 2018), can be created between the cloud and IoT systems. Performance analysis of the health data transmission of such systems were made and their usability was shown (El Kafhali and Salah 2018).

A study (Ghosh, Halder, and Hossain 2016), is based on obtaining the health data in the real environment over the IoT network using TCP protocol. However, there are no constrained devices in the study. The study focuses on obtaining health data rather than

communicating. In the literature, the benefits, vulnerabilities and solutions of the transmission of health data in the IoT environment are also discussed (Nausheen and Begum 2018).

A survey study (Khattak, Ruta, and Sciascio 2014), analyzes transferring health data with CoAP protocol within IoT scope. The study was focused on obtaining health data with the CoAP protocol and IEEE 802.15.4-6LoWPAN protocols without any performance evaluation. With the CoAP protocol, health data of patients can be monitored as web-based (Ugrenovic and Gardasevic 2016). In addition, there are studies suggesting the co-use of CoAP and HTTP protocols comparing the CoAP protocol with the HTTP protocol (Ge et al. 2016) that requires more system sources. The comparison of CoAP with MQTT, which can be used in instead of CoAP, (Imane, Tomader, and Nabil 2019), is also available in the literature.

There is also a comprehensive survey article (Islam et al. 2015) on the transmission of health data on IoT networks. The study focuses on subjects as; health sensors that can be used, meaningful health data, usage cases, different patient conditions, different network infrastructures, health data and transmission of health data in IoT environment. The study was not only limited to these reviews, but also examined other current technologies that could be used for health data, and detailed the policies of different countries on obtaining remote health data. In addition, problems that may be encountered are also presented. Another survey study (Qi et al. 2017), examines the transmission of health data in IoT environment from a different perspective. There are more research studies (Baker, Xiang, and Atkinson 2017; Dey, Ashour, and Bhatt 2017; Mainetti, Patrono, and Vilei 2011; Qi et al. 2015; YIN et al. 2016) in the literature.

The security of health data is also an important issue in this area. It is possible to transfer secure health data using enhanced DTLS with CoAP-based authentication (Kumar and Gandhi 2017). The DTLS protocol is a security protocol for UDP-enabled, constrained devices. Using CoAP messages, a handshake is performed with a certificate-key exchange. Together with connection security, patient data must also be confidential. For this reason, there are studies (He et al. 2018) about encrypting the private data and selection of passwords. LiBAC (Yang, Liu, and Deng 2018) is a proposed system on the privacy and security of health data in the IoT network environment.

One of the most interesting studies on the transmission of health data in the IoT environment is InLife (Koutsouris, Giannakopoulou, and Luca 2018). In this study, users with health data are expected to complete certain tasks as if they are playing a game. This

system, which uses scoring, encourages the user to do activities such as sports. The follow-up of the activities is done with the health data coming through the IoT network.

Studies in the literature show that the IoT is usable for health data transmission. In addition, various performance evaluations are available. However, in our knowledge, there is no study on the performance evaluation of the protocols on the 6LoWPAN infrastructure in the literature. In particular, there is no study that examines the MAC and RDC layer protocols, as far as we know.



4. MATERIALS AND METHODS

4.1. Materials

4.1.1. Software environment

As it is cost effective and provides ease of use and monitoring, a software-based simulation environment is preferred for measurement and evaluation. Although there are various network simulation environments, the number of simulation environments supporting the Internet of things is limited. Also, not all of them support constrained IoT devices. Other than that, some simulators do simulate only simple communications between nodes. Cooja (Österlind 2006; Sehgal 2013), the most widely used simulation environment in this field, was chosen as the default simulation environment as it meets the needs of this study.

Operating systems are available that can operate with the limited hardware of IoT devices and enable the use of hardware with a simple interface. These operating systems generally provide the same network protocol stacks. Choosing a common operating system that can work on IoT devices to be used in both hardware and software environments is important for the accuracy of the analysis. ContikiOS is an operating system that we can test and use in both software and hardware environments and its details are presented in the following sections.

In IoT devices used in software environment, the point is that the device has the desired hardware (like 802.15.4 radio) support. In the Cooja simulation tool that we mentioned and selected, there are IoT node devices that contain the desired hardware features. Among them, Zolertia Z1 and WisMote devices were preferred.

When choosing a CoAP client application to obtain data from the IoT network environment, a lightweight client with multi-thread support should be selected so that it does not affect performance analysis. The Californium CoAP client, which can provide these features and more, is the most prominent among many CoAP client applications.

Considering all these issues, the IoT environment, in which we can prepare a prototype in the real environment, was created in simulations and the analysis were made based on these simulations.

4.1.1.1. Cooja network simulator

Cooja network simulator is a network simulator tool running on java platform, developed primarily for contiki operating system. By creating a simulation environment, this tool can simulate many IoT devices and wireless communications between them. Some features of Cooja simulator is listed as:

- Simulates multiple types of nodes with full features.
- Has multiple modules such as node output to show serial interface of node or radio activity viewer to trace radio on time of a node.
- Has plug-in support for additional features.
- Can simulate some hardware sensors and equipment belonging to nodes.

The details of how this tool is configured for tests and how tests are performed are described in the methods section. A screenshot of the Cooja simulation tool is given in Figure 4.1.

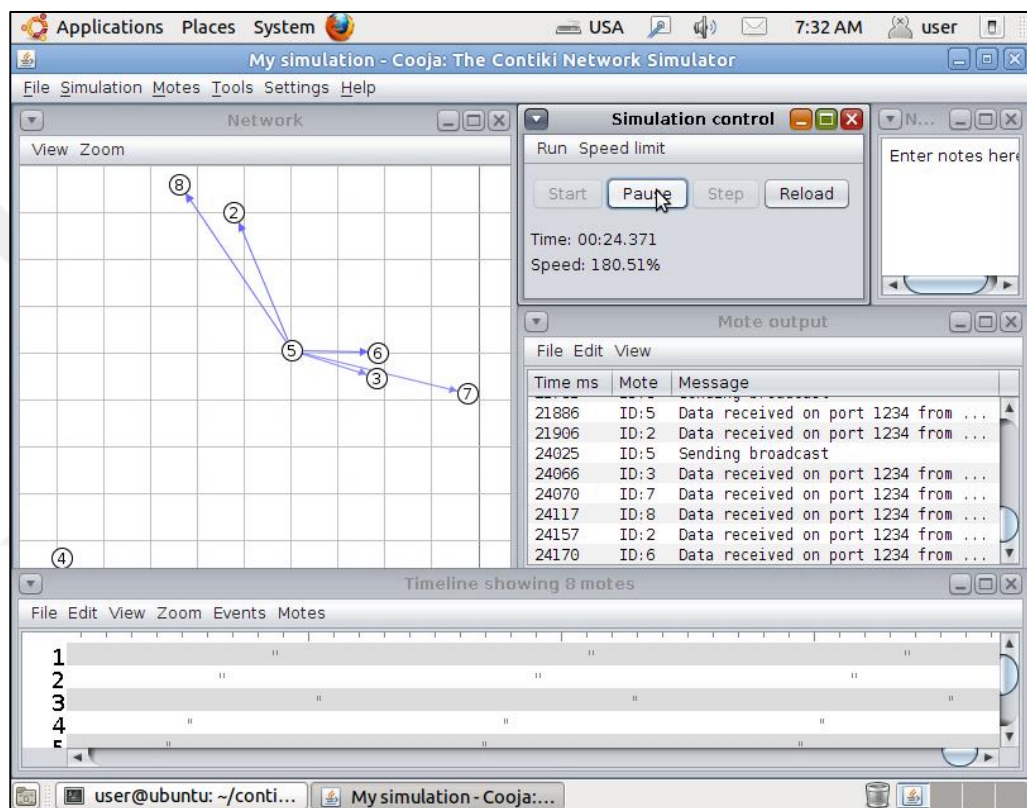


Figure 4.1 - Cooja Network Simulator

4.1.1.2. ContikiOS

While selecting an IoT operating system, there were multiple choices. ContikiOS (Dunkels et al. 2011), OpenWSN (Watteyne et al. 2012) and RiOT (Baccelli et al. 2013) are the most suitable operating systems for constrained IoT devices. All these operating systems are compatible with our current hardware environment. However, only ContikiOS has the full compatibility with Cooja, our selected simulation environment. Other than that, ContikiOS has more community support than other operating systems. Also, ContikiOS has network driver selection which allows us to change MAC, RDC layer protocols easily.

Although different operating systems are available for IoT networks, Contiki OS was chosen as it has more advantages than other IoT operating systems. Furthermore, it is

compatible with wide variety of IoT devices on both simulation environment and real hardware.

Contiki OS is used in simulations, as well as on real hardware, by installing it on nodes. The use of ContikiOS on the hardware environment is described in the relevant section. Features of ContikiOS is listed as:

- Full IP networking
- Memory allocation
- 6LoWPAN, RPL, CoAP protocol support
- Supports Radio Duty Cycling with contikiMAC protocol
- Allows MAC, RDC protocol changes
- Full Cooja simulator support
- Power efficient
- Open source

4.1.1.3. WisMote (Used in simulation)

In the Cooja network tool, the number of virtualisable nodes in which the contiki operating system can be installed and where the desired applications can be executed is limited. Zolertia z1 (left on Figure 4.2), which is one of these nodes, can run the desired applications and protocols with contiki operating system base. Modeled on real hardware, this virtual node has a 16-bit processor running at 16Mhz clock speed, just like the real model. This node is also equipped with 8 KB ram and 92 KB flash memory. Although these hardware features are insufficient in some cases for the desired environment. This was the most powerful mote type that could be added in the default environment. However, when the protocols with more memory consumption are run, the hardware of this mote is inadequate. Especially because of insufficient RAM memory, communication problems were encountered due to the "number of routes to be kept" which should be reduced. For these reasons, mote types with more RAM memory, which could be operated in simulation environment, were investigated.

In the default environment, WisMote (right on Figure 4.2), which cannot be added due to a bug in the 4.6.3 version of the MSP430-gcc compiler, has the desired properties for simulations. The error in version 4.6.3 of MSP430-gcc that makes it impossible for us to add this mote by default is due to incorrect entry of memory type of this mote type. As this issue was solved by version 4.7.0, the MSP430-gcc version to be used in the simulation environment was updated to 4.7.3. After this update, the node can be added to the simulation environment and can be run.

Although WisMote shares the same processor structure as the Zolertia Z1, it has 16KB of memory as RAM memory and 256KB of memory as its flash memory. These memory values are in amounts that can support the protocols that we want to run in simulations.

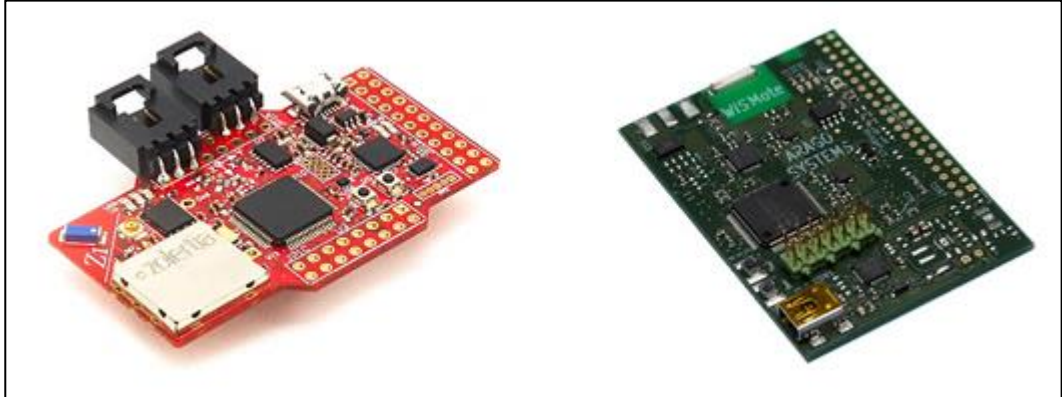


Figure 4.2 - Zolertia z1 (left) and WisMote (right) IoT nodes

4.1.1.4. Californium

Californium (Kovatsch, Lanter, and Shelby 2014) is one of many CoAP compatible clients and has been developed in java. Californium supports CoCoA and CoCoA+ (Betzler et al. 2015) application layer protocols which are improved versions of CoAP congestion control protocol. Broad protocol support, ease of use and detailed implementation were the reasons for choosing Californium. And the fact that Californium is written in java language brings multi-thread support together. In this way, Californium, which is already a lightweight application, can work as more than one client and will be able to take the results without any external effects.

To establish a link between Californium CoAP client and server node inside of IoT environment, a border-router mote should be placed. This border-router provides a connection between IoT network and Internet. Border-router mote is also known as gateway router as shown in Figure 1.1 in section 1.1. Through border-router, californium is able to send and receive CoAP messages (GET, POST, PUT, etc.) to/from any server mote inside IoT network.

4.1.2. Hardware Environment

While the measurements and evaluations made in the simulation environment are accurate, it is necessary to test these results on the actual hardware. In addition, sample health data, which we can use in simulation environments, was first obtained with real hardware environment. For this purpose, we selected the MySignals kit that we use to collect health data combined with OpenMote (Vilajosana et al. 2015) devices that supports

preferred operating system and IoT protocols. Detailed information and data about OpenMote and MySignals devices are given in next sub-chapters.

4.1.2.1. IoT device: OpenMote

OpenMote, which is selected as a hardware node, supports the desired operating system and the protocols to be used. OpenMote offers an open-hardware and open-software environment for faster development.

Although OpenMote nodes consist of multiple parts, it is the OpenMote-CC2538 module that takes over the actual job. As the name suggests, this module accommodates the Texas Instruments CC2538 (Instruments 2015) SoC. The TI CC2538 chipset features a 32-bit Cortex-M3 microcontroller and one IEEE802.15.4 radio. This microcontroller has 32 KB RAM and 512 KB Flash Memory and clock speed is up to 32 MHz. There are also GPIO, ADC, I2C, SPI, UART and timer modules within the microcontroller. The radio, the other part of the SoC, operates at a frequency of 2.4 GHz and is fully compliant with the IEEE802.15.4-2006 standard.

Another module for the OpenMote platform is the OpenBattery module, which provides power to the main module with the battery, as well as accelerometer, light, temperature and humidity sensors.

The OpenBase module, another part of the OpenMote ecosystem, is intended to increase the interfaces of the main module. The OpenBase module includes USB, 10/100 Mbps Ethernet and 10-pin JTAG connectors.

While the OpenMote ecosystem initially had different modules, in the new revisions, these modules (such as OpenBattery and OpenBase) were merged into a single main module. All OpenMote family products are listed in Figure 4.3.

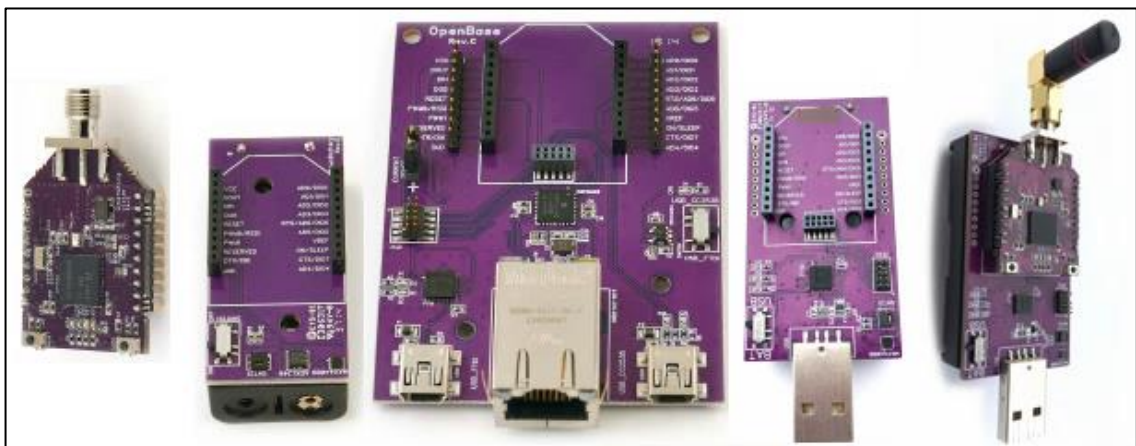


Figure 4.3 - OpenMote, OpenBattery, OpenBase, OpenUSB, OpenMote Rev.A1 (left to right)

4.1.2.2. Health sensors and other equipments

As hardware for health sensors, the MySignals (Figure 4.4) set has been chosen because of its ease of use and wide community support. It is possible to connect many hardware health sensors to this set, which basically uses the Arduino (Figure 4.5) microcontroller to read data from the sensors and process the read data.

The data read with MySignals Health Sensor kit can be transferred to external environment via Arduino serial port. Although this data can be read directly from the serial port, it was necessary to install a bridge element for serial port communication, as the hardware development equipment was limited. The device selected for bridging the serial port communication is the Raspberry Pi microcomputer. Since this device is not used for any purpose other than a bridge, it is not mentioned in detail. In real hardware development, it is possible to connect the MySignals kit directly to the IoT device by disabling the Raspberry Pi and Arduino equipment. However, this requires hardware re-design. Thus, we used Raspberry Pi as bridge between OpenMote and Arduino devices.

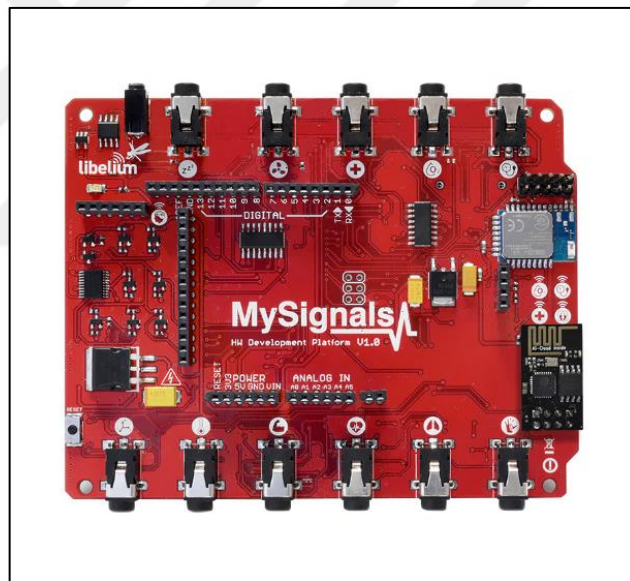


Figure 4.4 - MySignals HW Health Sensor Board

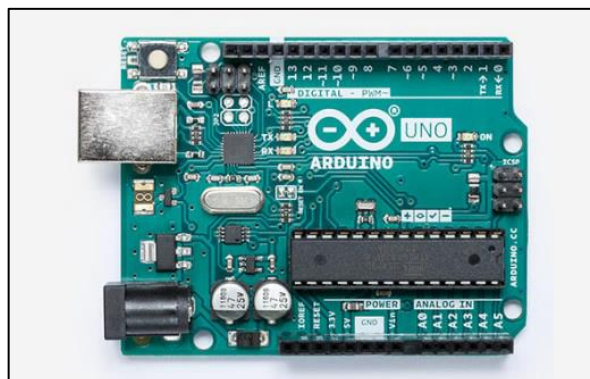


Figure 4.5 - Arduino UNO Microcontroller Board

4.1.3. Network Stack

IoT Network stack is a layered stack system like OSI model. As requirements of an IoT system differs from generic network models, its layer system is slightly different than OSI model. At the bottom of layered stack, link layer divided into 3 parts as Radio, Radio Duty Cycling and MAC, from bottom to top. Adaptation layer is composed from 6LoWPAN and IPv6 layers while IPv6 layer also responsible for routing. Transport layer exist on top of adaptation layer while it's beneath application layer. The layer structure and the protocols used in the layers are given in Figure 4.6 with the comparison of the traditional Internet stack model. In the image, the left-hand stack shows the LLN (Low Power and Lossy Network) Internet Stack, and the right-hand stack shows the standard traditional Internet stack model.

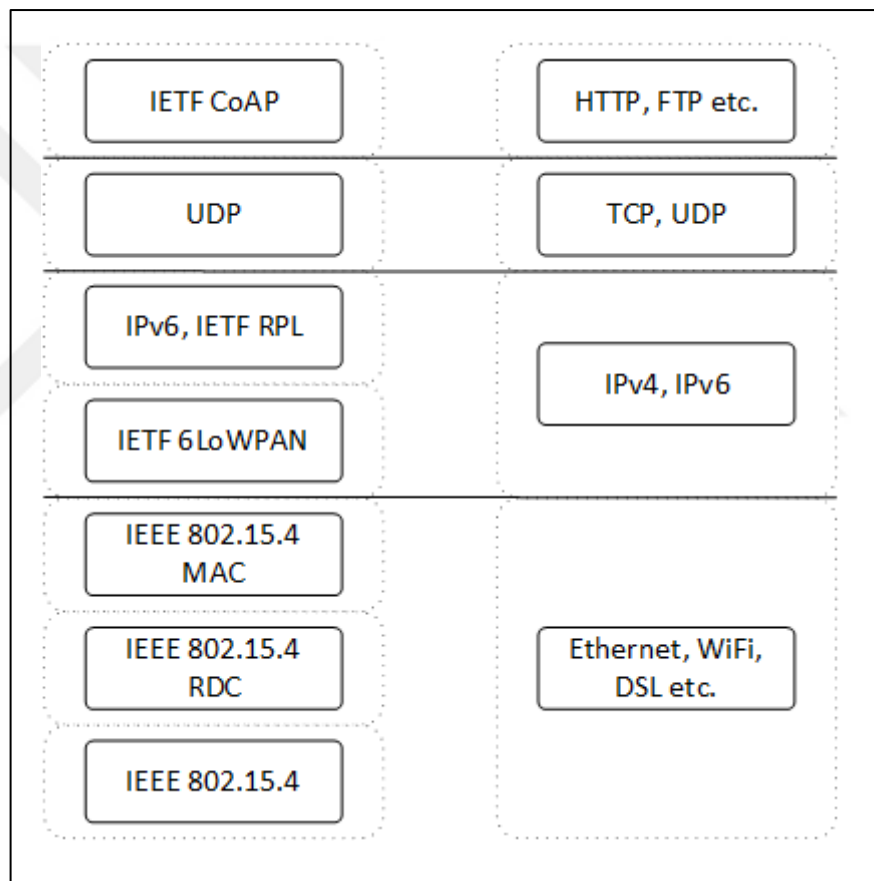


Figure 4.6 - Network Layers of LLN (left) and traditional Internet Stack (right)

4.1.3.1. Physical layer: 802.15.4 radio

IEEE 802.15.4 standard defines physical layer and media access control layer operations for low data rate devices. The radios comply with this standard uses three unlicensed frequency groups; 868.0-868.6 MHz for Europe, with single channel, 902-928 MHz for North America with up to 10 channels and 2400-2483.5 MHz for worldwide with up to 16 channels. Most IoT devices and simulation environments uses worldwide frequencies

with 16 channels. According to 16 channels, the channel layout is as in Figure 4.7. The 802.15.4 radio can offer a bandwidth of 250 kbit/s in practically around 10 meters range in general. Tradeoffs are possible to achieve lower energy requirements. Therefore 20 and 40 kbit/s transfer rates are also defined.

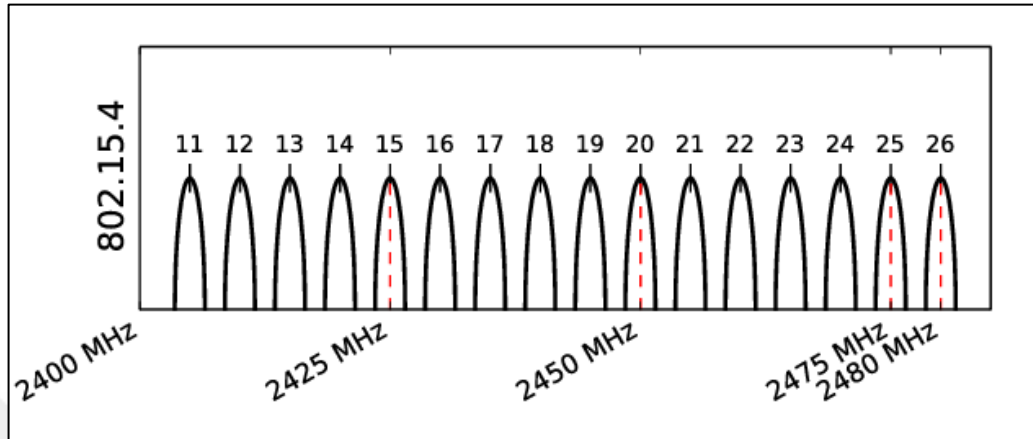


Figure 4.7 - 802.15.4 Radio Channels

4.1.3.2. Radio duty cycling (RDC)

On any environment, it's generally unpractical to change batteries. Especially if related environment is based on healthcare systems. For this purpose, there's some Radio Duty Cycle protocols for 802.15.4 based devices. Even if 802.15.4 radios power consumptions are low, this can be decreased more. Radio Duty Cycle approaches aims to turn off the radio as much as possible. Rendezvous points are set for nodes, at the beginning stage of network generally. Nodes turn on their radio on scheduled intervals to communicate and then turn their radios off after successful communication.

For un-scheduled communications, a node turns on its radio for a short time period to listen radio environment. If there's any activity on radio environment, it keeps the radio on in case of packet receiving. This technique is named as Low-power listening. If a node has no rendezvous point with any node, it uses Low-power probing. In this technique Receiver nodes turn their radios on and transmits a probe into radio environment periodically while powering on their radios a little longer. Sender node turns on its radio and listens for a probe and sends its packet as soon as it catches a probe.

Within Contiki OS there's 3 available RDC algorithms; ContikiMAC, XMAC and nullRDC. Since XMAC cannot be run on the cooja simulation tool, it is not included in the benchmarks.

ContikiMAC

ConitikiMAC is an RDC layer protocol that allows nodes to sleep when there is no communication. Thus, it is aimed that the nodes consume less energy. ConitikiMAC sleeps

the nodes in such a way that they wake up and listen to the line at regular intervals. If the node catches any packet transmission during its wake, the node remains awake. During the time that it is awake, it receives the packet that is in transmission, then a link layer ACK packet is sent to the sender. The node that wants to transmit a packet sends it continuously until it receives an ACK packet.

The ACK packet is not expected for broadcasts to be sent on all nodes. Instead, the packet is repeatedly sent over the entire waking time interval. In this way, the awakened nodes will receive the broadcast packet.

nullRDC

nullRDC is not actually a protocol, it refers to the absence of a protocol running on the RDC layer on Contiki OS. If no protocol is running in the RDC layer, the radio of the node will remain on continuously. In this case, it is expected that the node consumes more energy. Although it can consume much energy, it is unlikely that the packets sent will be missed by the fact that the radio is always on. In this respect, the lack of protocol in the RDC layer may affect the performance positively.

4.1.3.3. *Medium access control (MAC)*

MAC layer provides mechanisms to use the same radio environment without collisions. It backs-off if there's traffic on environment to prevent collisions. There's CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) and nullMAC available in Contiki OS. 802.15.4 MAC header can be up to 25 bytes.

CSMA/CA

With this protocol, firstly, the data to be sent is perceived to be in use. The node first listens to the environment and examines whether another node is transmitting in the same environment. If any other node is transmitting at that time, the node waits a certain time. When the line is detected to be empty, different actions are performed depending on whether RTS / CTS (Request to Send / Clear to Send) is used in the environment.

If the environment is an environment where RTS / CTS is used, first the RTS signal is sent and the CTS signal from the receiver is expected. If no RTS / CTS is used in the environment or a CTS signal is received (in an environment where RTS / CTS is used), the node sends the entire frame to be transmitted.

The sender node then waits for an acknowledgement message from the receiver that the frame is correct and the checksum calculation is correct. In the absence of a confirmation message, the node assumes that the packet collides with another packet sent. This causes the node to enter into the binary exponential backoff state, which will wait for a while before re-transmit.

nullMAC

nullMAC is a simple pass-through protocol, but it does not operate like a standard MAC protocol. nullMAC protocol just calls the appropriate RDC functions when necessary. nullMAC is not responsible for retransmissions or collision avoidance/detection mechanism. This may result in collisions to happen. The use of the nullMAC protocol can improve performance in networks with fewer nodes, but may adversely affect performance on more crowded networks.

4.1.3.4. Network: 6LoWPAN

In the IoT protocol stack, the network layer can be divided into two layers, 6LoWPAN and IPv6. 6LoWPAN is an IETF workgroup and an abbreviation of IPv6 over a low-power wireless personal area network. 6LoWPAN concept aims to enable the internet protocol to be applied to even the smallest devices. Nodes in 6LoWPAN that use IPv6 protocols require a border router to access outside of the local 6LoWPAN because they use header compression and different MAC layer protocols. This border router encapsulates 6LoWPAN packets into UDP packets.

The 802.15.4 frame size is 127 octets. After 40 octets of IPv6 and 8 octets of UDP headers, the available space is 79 octets. If the MAC header, which can reach 25 octets, is removed from this field, the user will have only 54 octets left. This applies to non-security scenarios. When security is used, the space remaining to the user will be further reduced. For a system that already has limited hardware, this is an undesirable situation. On the 6LoWPAN layer, IPv6 headers can be compressed. Also, in this layer, fragmentation management is also done. In the 6LoWPAN layer, the redundant fields of the IPv6 header are discarded and the rest is compressed. The resulting 6LoWPAN header can be reduced to 2, 12 or 20 octets according to the options used.

4.1.3.5. Network: IPv6

The number of devices connected to the Internet increases day by day and finds millions. As IoT devices are added to these numbers, it is inevitable that this number will increase exponentially. For this reason, IPv6 is the preferred internet protocol in IoT environments.

IPv6 is a protocol developed by ietf, considering the lack of IPv4 in addressing. IPv4 provides 32-bit addressing, while IPv6 can provide 128-bit addressing, so that every device on the internet is targeted to have a completely different address. Addressing is not the only advantage of IPv6 over IPv4. IPv6 can also eliminate the NAT problem with the addressing solution, provide tighter security measures, operate in constrained devices with compressed versions, provide more mobility and automatic address configuration. The IPv6 header structure, which differs from the IPv4 header structure, is given in Figure 4.8.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version				Priority Class								Flow Label																			
Payload Length												Next Header						Hop Limit													
SOURCE ADDRESS (128-bit)																															
DESTINATION ADDRESS (128-bit)																															

Figure 4.8 - IPv6 Header

4.1.3.6. Network: RPL routing

RPL (Routing over Low Power and Lossy Networks), defined in IETF RFC 6550, is a routing protocol specifically designed for IoT environment. RPL provides routing between nodes within an IoT environment. Also, routing between the IoT environment and the Internet environment is achieved by RPL routing protocol. RPL is based on distance vector and source routing. RPL works with the IPv6 protocol in the network layer.

Distance vector works by considering inter-node connections as a vector. In such routing protocols, topology changes should be periodically informed to the neighbors. The distance indicated here means the cost of reaching the next node. In the distance vector protocol, the distance value is calculated together with the direction. The concept of direction represents the place where the routed packet should be delivered. Each node keeps the cost of the distance to all other nodes in the network to the vector. It is preferred for IoT networks because of having less computational complexity, lesser message overheads and distributed approach.

There are 3 node types within scope of RPL routing; mesh, leaf, and feather. Mesh nodes can both route and forward traffic. Leaf nodes can only route its own traffic and not able to forward the traffic while feather node is only capable of forwarding traffic and not able to route its own traffic.

The network created by RPL is called DODAG network structure. The network structure with a root node without any loops is called as DAG (Directed Acyclic Graph). If the structure has only one root node, with paths ends with it, this structure is named as DODAG (Destination-Oriented DAG).

DODAG structure has 3 types of messages; DAO (Destination Advertisement Objects), DIO (DODAG Information Object) and DIS (DODAG Information Solicitations). DAO messages are transmitted in the upper direction along the DODAG structure. This process works until DAO messages are received by the root node. This informs the root

node about network topology. DIO messages, on the other hand, are used for discovering new nodes, to transmit configuration parameters and, communication. A node sends its own DIO message if it received a DIO message. Node adds information of rank value and link metric of the received DIO message, to its own DIO message. This allows node to choose its parent node. If any received DIO message has better rank value, that node which sends the DIO message is selected as parent node. DIS messages can be considered as neighbor discovery messages. A node which is not part of any DODAG structure sends DIS messages. If any neighbor is a part of DODAG structure, it sends DIO messages to determine its rank value. After that, the node joins the DODAG structure and is able to send DIO and DAO messages.

There are 4 values for RPL while managing the network; RPLInstanceID, DODAGID, DODAGVersionNumber, and Rank. RPLInstanceID is a unique identification number for RPL. DODAGID defines the root node while DODAGVersionNumber defines the version number as the name suggests. To make a network unique, RPL casts with RPLInstanceID, DODAGID, and DODAGVersionNumber. The Rank value refers to the distance of the node to the root node. The root node itself has the lowest rank value.

Nodes with RPL can work with storing or non-storing mode. A node running in storing mode holds its own routing table and sends DAO messages as unicast to the parent node according to this table. Nodes running in non-storing mode do not have routing tables. Nodes running in this mode send DAO messages unicast to the root node.

4.1.3.7. Transport: UDP

UDP (User Datagram Protocol) is the protocol used to move message packets called datagrams between two nodes. UDP is a connectionless protocol. In other words, it does not require a prior agreement for the packets to be sent. In this way, it aims to minimize the effort to deliver the packets. UDP has a header that specifies the checksum information and source / destination ports for data integrity. Since no handshake is made for the data to be sent, the data is not guaranteed to be transmitted. This makes UDP unreliable. Since the main purpose of the UDP protocol is to transmit the data to the target as soon as possible, the minimum network load and the minimum protocol weight are targeted. The UDP header size is only 8 bytes, as in Figure 4.9.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Source Port																Destination Port															
Length																Checksum															

Figure 4.9 - UDP Header

4.1.3.8. Application: CoAP

As in the OSI architecture, the application layer is the layer on which the application will perform the communication. The tools are available in this layer for programs to use the network. On restricted devices, applications that will run on this layer should consume system resources as little as possible. Although it is possible to run application layer protocols such as HTTP on restricted devices, it is more convenient to select alternative protocols because it consumes a lot of system resources. For this reason, the CoAP protocol is preferred in this layer, which works like HTTP, but consumes much less of system resources.

CoAP Congestion Control Mechanism

CoAP, defined in IETF RFC 7252, is an application layer protocol developed for constrained network nodes. Nodes using the CoAP protocol can communicate with each other. Coap is a simple low-overhead-sized protocol with multicast support. These features are very important for an IoT network. In this way, CoAP has proved to be an appropriate protocol for constrained devices. The CoAP protocol can run on almost any device that can run the UDP protocol. The CoAP protocol can also make improvements in energy consumption due to its low consumption of system resources. The CoAP protocol can also provide secure communication with DTLS.

CoAP messages consist of 4 types; CON (confirmable), NON (non-confirmable), ACK (acknowledgement) and RST (reset). When a reliable communication is requested, the client sends a message of type CON to the server until it receives an ACK-type message from the server. After the sent CON message, the client must wait the ACK message for a while. The client who cannot receive the ACK message during the expected time resends the CON message. When the default settings are used, the timeout time expands exponentially after each CON message. If the server is unable to process the CON message, it sends the RST type message to the client instead of ACK.

If the connection does not have to be reliable, the client can send NON messages to the server. The server receiving this type of message does not send the ACK message to the client. From the client's point of view, it is not known whether the NON message reached the server. Bu still, if the server cannot process a NON message, it can send the RST message to the client. Figure 4.10 shows how CON and NON message types work.

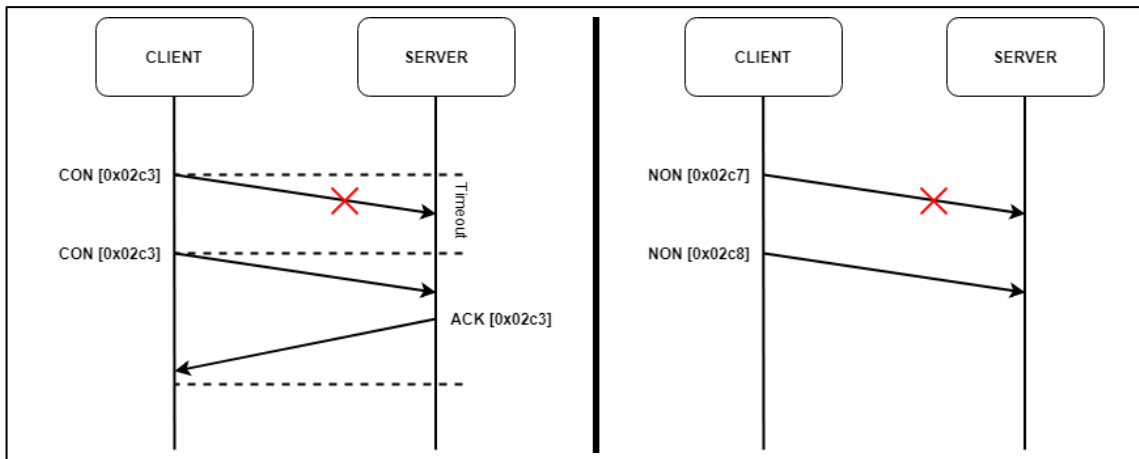


Figure 4.10 - CoAP CON (left) and NON (right) Message Types

CoAP congestion control uses binary exponential back-off mechanism. If a CoAP message does not have a response within timeout value, it should be retransmitted. Also, timeout value is doubled when a retransmit occurs. Aforementioned timeout value is configurable by client-side application and it should be based on RTT.

There are many different CoAP applications written in different programming languages. With most open source applications, the use and learning of the CoAP protocol has been made quite easy.

4.2. Methods

4.2.1. Obtaining health data

The health data in question are not data that can be obtained in the simulation environment. For this reason, the equipments mentioned in the Materials section are used to obtain health data. These equipments, in which health data is collected, works with the Arduino microcontroller interface, as mentioned earlier. Although there are code samples to run on the arduino microcontroller for these health sensors, there is no comprehensive application code. For this reason, an arduino program was written to read the sensors for all health data to be obtained. The sensor data obtained may be of different data types. For this reason, all data must be converted into a single format when preparing the data. The health data prepared were encoded on the arduino in accordance with a specified bit sequence, and the data was sent from the serial interface on the arduino.

Once the MySignals device we use to collect health data with the Arduino microcontroller is ready for use, this health data should be transferred to the IoT device. The OpenMote node, the IoT device we use, does not have an interface to communicate directly with the Arduino microcontroller. For this reason, a Raspberry Pi microcomputer was installed between the two devices, which would serve as a bridge. With a few hardware modifications to the OpenMote node or Arduino microcontroller, the need for a Raspberry

Pi microcomputer can be eliminated. Since hardware development is not the subject of this study, this issue has not been discussed. A sample setup prepared with several health sensors is as in Figure 4.11. Arduino and OpenMote devices are connected to USB ports on Raspberry Pi. On these USB ports, they can send and receive data via a virtual serial interface.

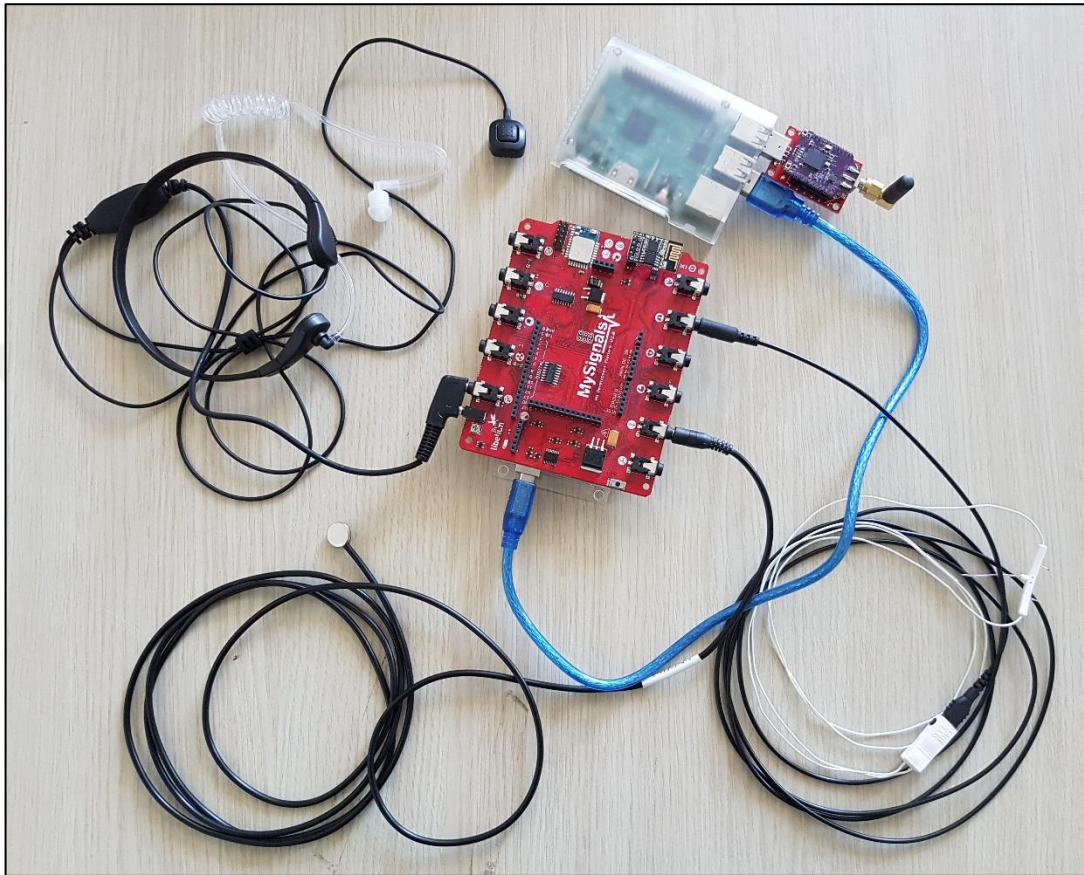


Figure 4.11 - Hardware Setup Used to Collect Health Data

With this setup, the data was tested by reading on the Raspberry Pi microcomputer before being sent to the IoT device. After making sure that the data is seamless, the bridge software has been prepared to communicate the virtual serial interfaces created on the USB ports of the Raspberry Pi microcomputer. This application, written using NodeJS, communicates the virtual serial port connected to OpenMote with the virtual serial port connected to Arduino. After establishing a connection between Arduino and OpenMote, the IoT device (OpenMote) must transmit the health data received to it via the 802.15.4 radio. At this stage, for the ContikiOS operating system running on OpenMote, an application was written in C to read the serial interface and transfer the incoming data over the network. At this stage, the setup is ready to collect health data and forward it to the IoT network. This setup can be called as a server node.

The node that will be in the Gateway role consists of the Raspberry Pi microcomputer and the OpenMote IoT node. The gateway node is responsible for collecting health data, processing health data, and connecting the IoT network to the Internet network. With the applications to be run on the Raspberry Pi microcomputer within the Gateway node, data can be collected from the gateway node IoT device. Through this node, any connection to any other node (the node that collects the health data) can be considered as a client. This setup can be called as a gateway node. Within this setup, we can call each application piece that collects data as a client.

After a server node and a gateway node were created, the client application was run on the gateway and the current health data was obtained. At the same time, the health data is also monitored via the server node (before being transmitted to the IoT network). Two health data were compared and no difference was found between them. In this way, it is checked that the IoT network transmits health data without error. As a result of these processes, both the collection of health data in the real environment has been tested and the sample health data which can be used in simulations are obtained.

4.2.2. Determination of vital health data traffic characteristics

Data from health sensors are mostly numeric data in float or integer format. These data should be minimized as the intended purpose is to transport these data as soon as possible and accurately. For this reason, the maximum and minimum values that these health data were determined first. After each sensor data was examined in this way, a bit sequence was created by allocating as much space as needed for each sensor data. The result is a 64-byte bit sequence containing all sensor data and the free space to which additional data and options can be added.

In this model, there are 11 different sensors with 21 types of data in total. As a sensor can deliver more than one kind of data, a difference occurs in this way. Most sensor data can be expressed in the form of unsigned integer data. Long unsigned integer and byte data types are used for data that does not fit the unsigned integer data type or is smaller. Of the 21 data types used, 1 is a long unsigned integer, 1 is a byte, and the remaining 19 are unsigned integer. In this structure, the total data length is 344 bits (43 bytes). The remaining 168 bits (21 bytes) of space may be filled with patient information, other sensor data that may be added, or other options. These data types and lengths are given in Table 4.1.

ID	Sensor	Bytes
1	Body Position	6
2	Body Temperature	2
3	Snore Sensor	3
4	Galvanic Skin Response	8
5	Airflow Sensor	2
6	EMG Sensor	2
7	ECG Sensor	2
8	SPo2	4
9	Blood Pressure Sensor	6
10	Body Scale	2
11	Glucometer	6
12	Patient ID	1
	Total	44
	Free Space	20

Table 4.1 - Health Sensors and Data

Although 64 bytes of data packets appear to be small, they can contain many health data without compression. In future studies, these 64 bytes can be used more effectively by optimization and compression methods.

It is also possible to send all these health data as separate messages for different types of health data. However, sending large, unified health data is less costly than sending these data separately. For this reason, it is more logical that all data are transmitted as a single unified structure.

Patients to collect health data may have different conditions. In this case, the needs of patients in different categories may be different. What is required here is the health values that need to be measured. Therefore, patients were divided into three categories according to the sensors to be measured. The patient category, which requires continuous and immediate observation of health status, are critical patients. Patients whose condition is not critical, but need to be monitored and measured at regular intervals, can be categorized as non-critical. In addition to these two categories, individuals who are not in any health status (critical or non-critical), but are followed-up for probabilities are examined under the follow-up category. Critical and non-critical patient categories are more suitable for hospitals, clinics, nursing homes, while the follow-up category is suitable for nursing homes and child care homes.

In addition to categorizing patients, health sensors can also be categorized according to needs groups. For example, for a patient who is permanently lying down, body position sensor data may not be important, while the same sensor may be important for an elderly patient to detect events such as falls. Health data that may be important for the

patient should be measured more frequently. For such reasons, for each category of patients, the sensor data is divided into categories according to their importance (and hence the frequency of measurement). The sensor data is divided into 4 categories according to the measurement frequency requirements. Patient categories and measurement frequency requirements based on the information obtained from doctors are listed in Results section **Hata! Başvuru kaynağı bulunamadı..**

The numerical data indicated in the table indicates how many minutes the relevant sensor should be updated at the latest. For example, Airflow, ECG, EMG, Spo2 sensors require almost continuous measurement for critical patient categories. However, the Galvanic Skin Response, Body Scale and Glucometer sensor data is sufficient to take one hour.

4.2.3. Preparation of simulation environment

4.2.3.1. Cooja network tool & contiki os

As mentioned earlier, the installation and operation steps of the cooja tool, which is determined as a simulation tool, are indicated in this chapter. The cooja simulation tool works on the Ubuntu operating system in the most efficient way. For this reason, a virtual computer was created on the computer using VMWare, and Ubuntu 16.04 operating system was installed on this computer. After the required application packets were installed on this operating system, the contiki operating system development folder, including the cooja network tool, was downloaded via github. Then, additional packets that the contiki operating system or cooja tool might need were downloaded and installed. Detailed steps on these procedures can be found on the contiki operating system website and related forums.

After the Contiki operating system and the cooja network tool are operational, the cooja tool is started. In this step, the virtual environment of the simulation will be displayed. In this step, a contiki operating system with the desired application will be installed to the virtual node wismote mote and the node will be added to the environment. The first node to be added must contain the border-router software and must be in the border-router function as appropriate for the software in it. In addition, the serial interface that enables the border-router node to communicate with the out-of-simulation must be activated for this node. The Border-router node, as mentioned earlier, is the node that will allow our IoT network to access the Internet. The environment in which the simulation environment operates The IoT network can be thought of as the virtual computer itself (which runs the Ubuntu operating system) as an internet network. Since the connection runs from the serial interface, there is a need for a bridge connection outside the simulation environment that can connect the simulation environment with the computer. This tool is offered within the contiki operating system. A virtual interface is created with the Cooja-border-router tool to create a bridge

connection between the simulation environment and the actual computer. After this step, the network connection between the main computer and the simulation environment is provided.

In the next step, the wismote nodes that have the software installed to transmit the health data from the sensors to the network environment are added to the simulation network environment. The desired number of nodes and network topology are visually established in the simulated environment. From the options, the simulation speed should also be selected 100% for a realistic approach.

Energy efficiency, which is one of our performance metrics, can be calculated approximately by means of the cooja tool with the sleeping times of the radios of the nodes. These values (Radio Duty Cycle values) can be displayed with the powertracker plugin in the cooja tool. Other performance metrics cannot be read directly from the cooja tool. Other performance metrics can be obtained by receiving data from a client reading data from these nodes responsible for delivering health data. This client is the Californium CoAP client, the details are given in the next section.

4.2.3.2. Californium coap client

The Californium application described in the Material section has been chosen as the default CoAP application in the tests performed. Californium is an open source application that can be downloaded from github.

Based on the hello-world application in the Californium downloaded from the Github packets, a benchmark application has been developed. The new application was designed to be based on the multi-client model. For this reason, the application is designed and developed as multi-thread. The multi-threaded application makes it possible to run a client for each server. Thus, the result values to be taken will be independent of the nodes and will not affect each other.

Latency, Reliability and Throughput performance metric values are gathered by creating a separate Californium CoAP client thread for each server. Once the values are obtained, they are formatted to form a meaningful graph.

4.2.4. Performance metrics

Some performance metrics have been identified in order to determine the extent to which the collected health data is useful and to measure the performance at the stage of obtaining useful data. Some of these metrics have helped us to determine whether the resulting health data were useful. Some metrics have enabled us to perform performance analysis of different network stacks. These metrics and their descriptions are explained in order of importance.

4.2.4.1. Latency

Since the study involved critical health data, it should be ensured that these data are delivered to the corresponding point as soon as possible. Latency metric tells us how long it takes delivering packets to destination. This metric represents the maximum delay time between two successful packets. Thus, the knowledge of how long the health data is renewed at the latest is obtained. In the study, patients were categorized according to the importance of the sensors to be measured. Similarly, health data are also divided into groups according to their importance in patient categories. Each health data may require different rates of renewal for different patient groups. For example, ECG measurements of a critical patient should be performed at frequent intervals. For this reason, this metric will be used to determine in which scenarios the health data will be used in the patient groups. Patient categories and measurement frequency requirements based on the information obtained from doctors are listed in the Results and Discussion section.

4.2.4.2. Energy efficiency

While transporting vital health data, it is important that the system can continue to work. In order to achieve this, the energy needs of the systems that collect and transmit health data should be provided. These energy needs should be met with batteries as a wireless system is considered. As well as the health data collection and transmission systems to be used, the battery should be as light and portable as possible. In addition, undesirable situations such as frequent battery replacement should be minimized. In order to achieve this, power consumption must be kept to a minimum. Since the scope of this work is on finding the best environment for transporting health data, energy efficiency has been selected as an important performance metric. This metric can provide us with information about how long a node can work. Energy efficiency metric is calculated by total radio-on time. The WisMote used in the simulation environment and the OpenMote nodes used in the real environment use the TI CC2520 and TI CC2538 radio chips respectively. The hourly energy consumption data obtained from the data sheets of these radio chips are given in Table 4.2.

Power Consumption (mA)		
Mode	Radio IC	
	TI CC2520	TI CC2538
Active RX	22,3	20
Active TX	25,8	24
Sleep Mode	0,03	0,0013

Table 4.2 - Power Consumption of two different 802.15.4 Radio ICs

4.2.4.3. Reliability

One of the metrics required for the transmission of health data is reliability. One or more health data packets should not be lost during transmission. The fact that samples of health data can be taken more will prove the accuracy of the measurement. While this metric is not as critical as latency or energy efficiency, it will directly affect the transmission performance of health data. The measurement of this metric is calculated by the rate at which the packets sent are successfully transmitted to the recipient. The high transmission success rate will show the performance of the relevant environment / scenario.

4.2.4.4. Throughput

Regarding the transmission of health data, it is wrong to think that there will be data from a single patient on the system. It should also be possible to have simultaneous data from multiple patients. In this case, throughput is an important performance metric. This metric is calculated based on the sum of the packet sizes successfully transmitted over the network. Larger throughput is better because it means more data can be carried on the network which means support for more patient data. With the data of this metric, the total throughput of the network or the average throughput of the nodes can be calculated.

4.2.5. Network topology

While the network topology is planned within the scope of the study, the environment such as hospital or nursing home has been considered. It is assumed that each patient is in a different room and nodes that transmit health data are fixed or move in a narrow space. In this structure, patients are located in 6x5 separated rooms with around 40 meters between them. From these patients, health data were considered to be transmitted to a root node at the endpoint. An example of the respective topology is given in Figure 4.12.

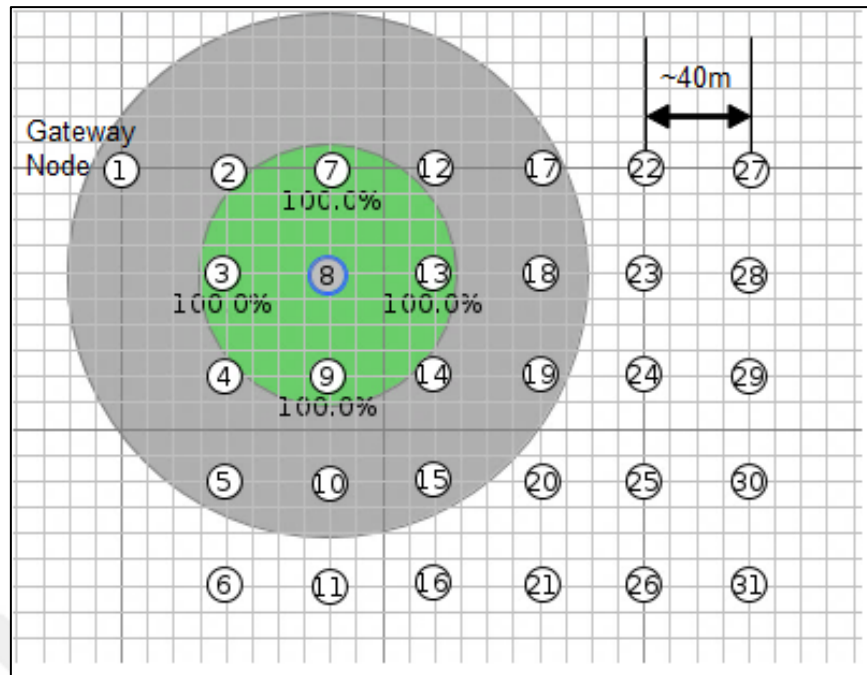


Figure 4.12 - Network Topology

In the Cooja network simulator environment, there is no transmission after 6 nodes in linear topology. For this reason, designed network topology is prepared in accordance with this limitation.

The traffic flow in the environment is from all nodes to the root node. Request messages sent by the root node are assumed to be insignificant due to their small size. While transmitting the health data, each node transmits health data packets to the parent node to deliver it to the root node. Also, each node (if its a parent of another node) forwards the received health data packets towards the root node through their parent nodes.

5. RESULTS AND DISCUSSION

5.1. Latency

Within the scope of the study, first of all, the usability of different health sensors for different patient categories was examined. In order to achieve this, the health data groups needed by each patient category were examined considering their latency values. As mentioned earlier, the required health data should be obtained at a certain time interval. That is why the latency metric is chosen as the distinctive metric. Before comparing the performances of different scenarios, the patient categories and health data for which the relevant scenario could be used were separated.

Sensor	Patient Category		
	Critical	Non-Critical	Follow-up
Body Position	30	30	15
Body Temperature	15	30	60
Snore Sensor	30	60	60
Galvanic Skin Response	60	60	60
Airflow Sensor	30	30	60
EMG Sensor	1	60	60
ECG Sensor	1	60	60
SPo2	1	60	60
Blood Pressure Sensor	30	60	60
Body Scale	60	60	60
Glucometer	60	60	60
Patient ID	-	-	-

Thresold 1: 1 minute
Thresold 2: 15 minutes
Thresold 3: 30 minutes
Thresold 4: 60 minutes

Table 5.1 - Health Sensor Data Acquisition Interval Thresold Values (in minutes) by Patient Categories

Table 5.1 lists the sensor data needed by the patients divided into categories. The data in this table indicate that the importance of the data required by different patient groups also varies. In this way, it may be necessary for the patient group to have a higher refresh rate for a health data. In this table, with the help of the information obtained from the doctors, the health data is divided into different refresh rate intervals according to their importance. The health data specified by red color are the data that require the most frequent refresh rate in critical patient groups. Therefore, the refresh rate of these data is determined as one

minute at the latest. Secondary health data is shown in orange color and the regeneration interval is determined as maximum 15 minutes. Examples of this type of data include measurements of the patients under follow-up to detect body positions (for detecting events such as falls) or fever measurements of critically ill patients. The health data in the third category, which is indicated by yellow color, means the health data that should be renewed with a maximum delay of 30 minutes. The fourth and last health data category is expressed in green color. These data are relatively less important. Therefore, the measurement frequency is defined as 60 minutes.

MAXIMUM DELAY BETWEEN 2 PACKETS												
PDR / Client	nuliMAC / nuliRDC			CSMA / nuliRDC			nuliMAC / contikiMAC			CSMA / contikiMAC		
	100	95	90	100	95	90	100	95	90	100	95	90
2	0,23	0,35	1,05	0,03	0,05	0,05	0,14	0,86	1,97	0,11	0,76	0,44
4	0,32	2,65	5,09	0,08	0,06	0,10	1,04	4,86	9,47	0,49	1,07	0,81
6	0,66	2,43	10,13	0,34	0,28	0,33	1,84	8,99	13,95	2,79	2,04	2,47
8	0,59	2,15	12,22	0,21	0,46	0,32	1,72	7,65	12,00	2,05	2,38	5,26
10	0,93	3,50	8,24	0,64	0,32	0,52	5,31	11,74	16,89	4,82	4,16	4,95
12	1,24	4,10	10,72	0,47	0,38	0,49	3,68	11,71	19,56	6,13	4,77	6,56
14	0,87	4,36	7,44	0,57	0,52	0,62	7,17	19,76	20,06	9,61	6,41	8,46
16	1,26	3,22	9,90	0,91	0,74	0,78	9,91	12,16	27,31	10,40	13,47	12,23
18	1,55	3,70	10,90	0,81	0,66	1,55	12,04	21,21	25,85	11,32	12,42	18,98
20	1,99	3,75	13,86	1,04	0,95	0,87	12,77	16,78	34,75	11,65	16,11	19,19
22	2,33	4,91	13,18	1,11	1,12	1,27	14,85	17,10	37,44	19,53	21,13	23,75
24	2,05	5,54	12,10	1,19	1,32	1,34	17,89	25,26	37,07	27,25	21,28	26,38
26	2,97	5,34	15,58	1,40	1,23	1,49	19,23	21,70	44,05	24,76	34,99	35,78
28	2,44	5,48	14,88	2,02	1,30	1,44	17,62	26,63	46,70	42,62	41,01	40,18
30	2,75	7,07	19,22	1,64	2,91	1,61	17,99	33,90	37,06	35,29	34,98	52,58
Thresold 1: 1 minute						Thresold 2: 15 minutes						
Thresold 3: 30 minutes						Thresold 4: 60 minutes						

Table 5.2 - Maximum Delay Between Packets (in minutes) in different scenarios

According to the data obtained from the simulation results, maximum delay between two packets values in different scenarios are given in Table 5.2. In the table, four different scenarios are presented with different PDR (Packet Delivery Ratio) values. Up to 30 clients were used in the simulations and consequently in the results table. In this table, according to the minimum threshold values, the appropriate data is colored according to the threshold values. An appropriate threshold value also applies to all threshold values greater than itself. For example, the data indicated in red represents 1 minute with the lowest threshold range. Since this threshold range is less than 15, 30 and 60 minutes with other threshold ranges, it is suitable for all patient groups and health data. The results can also be analyzed

with the graphics obtained for 4 different scenarios (different MAC and RDC layer protocols).

In Figure 5.1, the result graph of the scenario using nullMAC and nullRDC is given. The result graph of the scenario using CSMA and nullRDC is given in Figure 5.2, while the result graph of the scenario using nullMAC and contikiMAC is shown in Figure 5.3. Finally, the result graph in which the CSMA and contikiMAC protocols are used is in Figure 5.4. Latency threshold indicators based on Table 5.1 are also shown graphs.

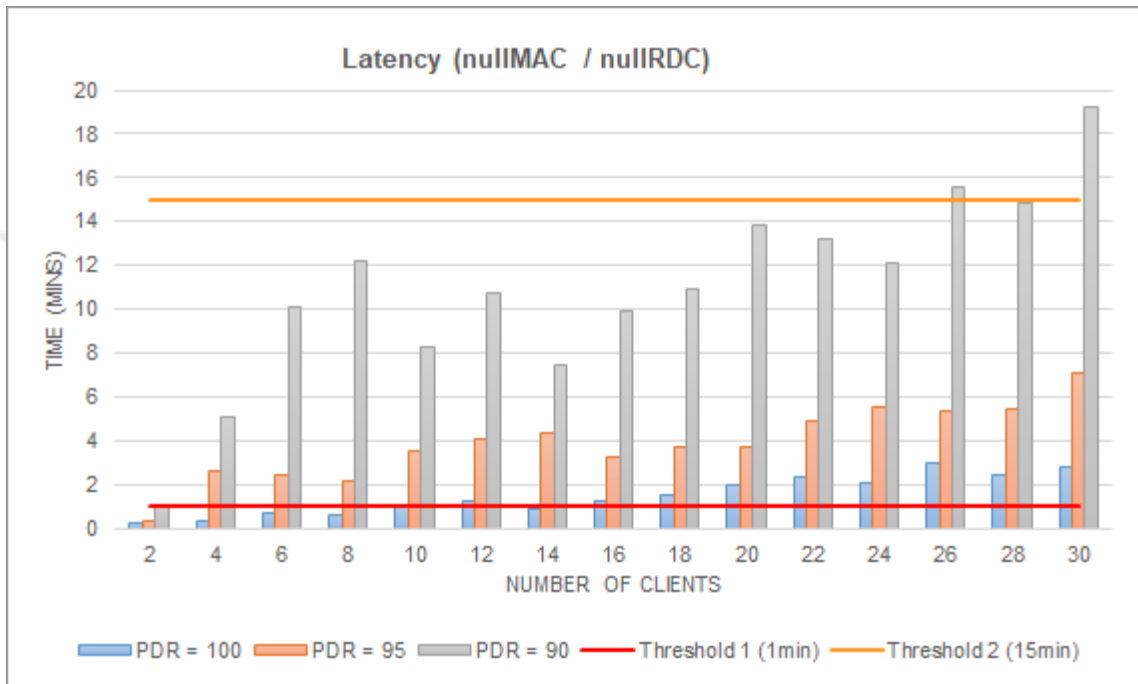


Figure 5.1 - Latency (Maximum Delay Between Two Packets) graphic of nullMAC / nullRDC scenario

In Figure 5.1 latency values are given for protocol stack with nullMAC protocol in MAC layer and nullRDC protocol in RDC layer with PDR values 100,95 and 90. In this graph, only thresold 1 and thresold 2 values are indicated. As there's no result to exceed or close to the thresold 3 (30min) or thresold 4 (60 min), these thresold values are not indicated. When the results are examined, in general, the reduction of the pdr value (decrease in the packet delivery rate) leads to increase in latency. When we look at the graph more widely, it is seen that the latency values of nullMAC / nullRDC protocol combination are usually less than 15min (thresold 2). However, there are also cases where the latency values are less than 1 min (thresold 1) and above 15 min (thresold 2). If the results are analyzed on average, it can be said that the combination of nullMAC / nullRDC protocol can be suitable for situations where the latency tolerance is less than or equal to 15min.

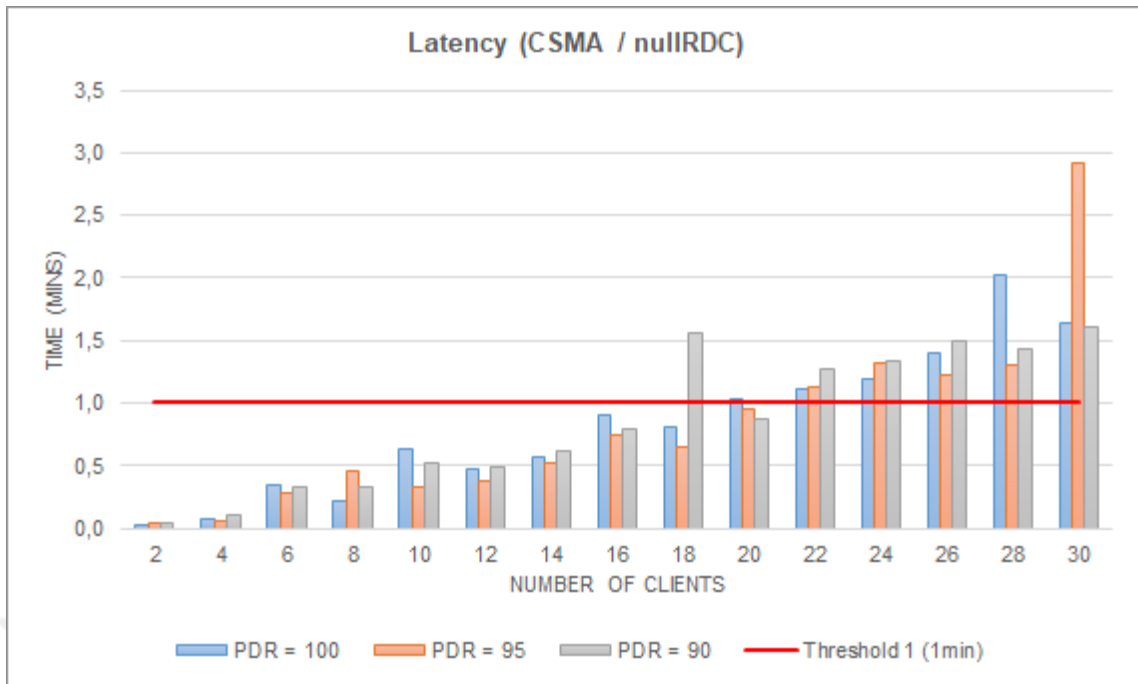


Figure 5.2 - Latency (Maximum Delay Between Two Packets) graphic of CSMA / nullRDC scenario

Scenario results for latency metric with using the CSMA and nullRDC protocols are given in Figure 5.2. As can be seen from the graph, when the number of nodes is 16 or less, the latency values are under threshold 1 (1min). Nevertheless, as a result of the increasing number of nodes, the latency values for all PDR values are gradually increasing. However, in all the results, the combination with the lowest latency values is the CSMA / nullRDC protocol combination.

The latency metric results in Figure 5.3 belong to the nullMAC and contikiMAC protocol combination. The latency values in these results are higher than the results of the protocol combinations we have previously examined. The significant increase in the latency values in this combination might be due to the contikiMAC protocol, which sleeps the radio to conserve energy. However, the latency values usually remain below threshold 3 (30 min) when the PDR value is low. In almost all PDR values and number of nodes, the latency values are above threshold 1 (1min). According to the results, when the number of nodes does not exceed 18, this protocol combination is suitable for threshold 3 (30min). In cases where the number of nodes is greater, this configuration will be compatible only for threshold 4 (60min).

The last combination of protocols, CSMA / contikiMAC, is given in Figure 5.4. This combination gave similar latency values to nullMAC / contikiMAC combination. This combination is compatible with threshold 3 (30min) in network scenarios with up to 24 nodes. This combination is not suitable for situations where the threshold value should be less than 1 minute (threshold 1).

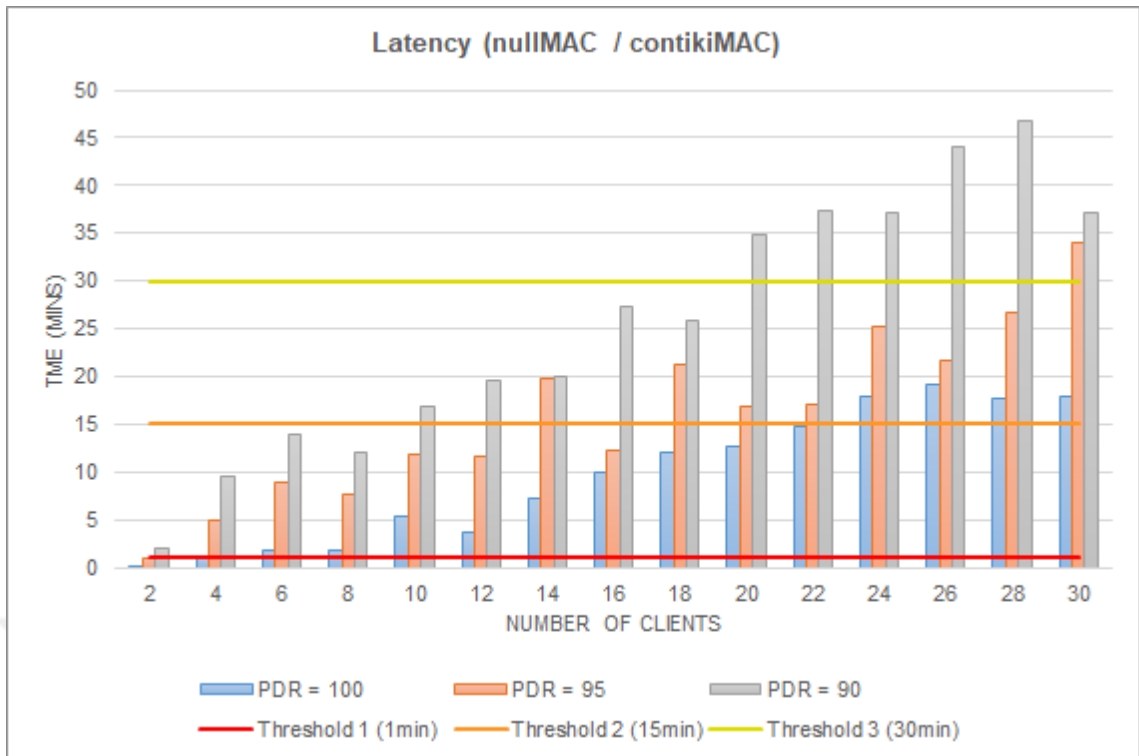


Figure 5.3 - Latency (Maximum Delay Between Two Packets) graphic of nullMAC / contikiMAC scenario

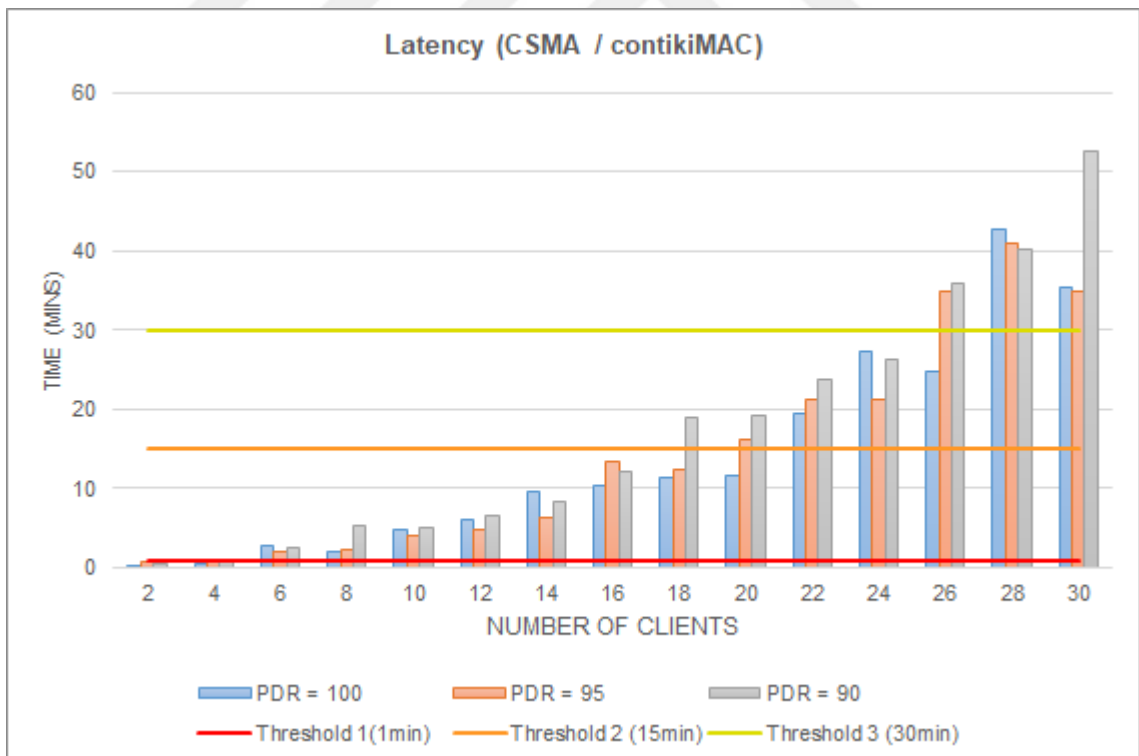


Figure 5.4 - Latency (Maximum Delay Between Two Packets) graphic of CSMA / contikiMAC scenario

According to the data in the table and graphs, it is clear which categorized health data can be transmitted in different scenarios.

5.2. Energy Efficiency

When collecting health data, the sensors and equipment on the patients should be as light and comfortable as possible. To achieve this, the batteries, which are the heaviest equipment, are minimized as much as possible. Naturally, small batteries will be weaker in capacity. The consequence of this is that the frequency of battery change increases. Just as heavy equipment is undesirable, frequent battery change is undesirable for patients. Therefore, the second metric that is important is energy efficiency. If the equipment used is designed to use the minimum energy, long working times can be achieved with light batteries. As the focus of the study was not on the health sensors themselves or on microcontrollers, this energy saving was tried to be provided on IoT devices.

In IoT devices, the most energy consuming unit is the 802.15.4 radio itself. As a solution to this, RDC protocols have been proposed to ensure that the radio is switched off when not in use. Among these protocols, contikiMAC, one of the most commonly used, is the RDC protocol used in simulations in the study. WisMote was used in the simulation environment. The radio integrated circuit on this node is the TI CC2520 model. The OpenMote nodes we tested in the real environment have a TI CC2538 radio integrated circuit. The energy consumption values of both radio integrated circuits are indicated in Table 4.2 in the previous Section: 4.2.4.2. These values were obtained from the data sheet of the respective radio integrated circuit. The active energy consumption of the active RX and active TX modes is different. There will be an RX node across the network, corresponding to each TX node. Therefore, the average of these values was used for the times when the radio was switched on. The energy consumption data in Table 5.3 were obtained according to these energy consumption values and radio on / off times.

ENERGY CONSUMPTION (mA per hour)												
PDR / Client	nullMAC / nullRDC			CSMA / nullRDC			nullMAC / contikiMAC			CSMA / contikiMAC		
	100	95	90	100	95	90	100	95	90	100	95	90
2	0,17	1,00	1,63	0,06	0,08	0,09	0,02	0,04	0,07	0,02	0,03	0,09
4	0,38	1,61	2,87	0,18	0,15	0,17	0,03	0,11	0,10	0,06	0,10	0,07
6	0,57	2,24	3,02	0,22	0,18	0,34	0,05	0,09	0,13	0,10	0,17	0,21
8	0,44	2,26	3,03	0,32	0,25	0,41	0,06	0,13	0,13	0,16	0,19	0,18
10	0,43	2,45	2,89	0,35	0,43	0,49	0,05	0,15	0,12	0,10	0,21	0,29
12	0,55	2,51	2,95	0,58	0,60	0,73	0,08	0,15	0,12	0,18	0,24	0,35
14	0,58	2,80	3,25	0,60	0,70	0,85	0,06	0,16	0,12	0,17	0,28	0,33
16	0,75	2,96	3,20	0,75	0,74	0,85	0,09	0,15	0,10	0,25	0,26	0,42
18	0,95	3,14	3,06	0,86	0,98	0,85	0,08	0,13	0,09	0,24	0,37	0,38
20	1,11	3,28	2,83	1,00	1,21	1,15	0,08	0,13	0,07	0,32	0,41	0,43
22	0,96	3,35	2,75	1,01	1,00	1,07	0,12	0,15	0,07	0,29	0,40	0,40
24	1,23	3,50	2,74	1,18	1,02	1,40	0,15	0,13	0,07	0,41	0,40	0,35
26	1,33	3,11	2,73	1,50	1,31	1,55	0,11	0,14	0,07	0,36	0,41	0,29
28	1,27	3,25	2,58	1,30	1,27	1,52	0,14	0,13	0,07	0,29	0,29	0,30
30	1,50	3,31	2,21	1,38	1,43	1,59	0,12	0,11	0,06	0,31	0,30	0,28
AVG	0,81	2,72	2,78	0,75	0,76	0,87	0,08	0,13	0,09	0,22	0,27	0,29

Table 5.3 - Average Energy Consumption per Node

The energy consumption in the table is indicated in mAh and refers to the average energy consumption of the nodes. Scenarios where at least one or more of the nodes cannot send data for 60 minutes have been highlighted in gray and underlined. As can be seen from the data, the use of the RDC protocol significantly reduced energy consumption, as expected. Charts based on the table are given below. The result data can also be examined with these graphs.

Graphics are grouped under different PDR (Packet Delivery Ratio) values. The values where the PDR value is 100 are given in Figure 5.5, the values where the PDR value is 95 are shown in Figure 5.6 and the values in which the PDR is 90 are given in Figure 5.7. In these graphs, it is shown how much energy is spent to send the packet with the protocol stack used. In addition, the graph containing the energy consumption average of the nodes is added in Figure 5.8.

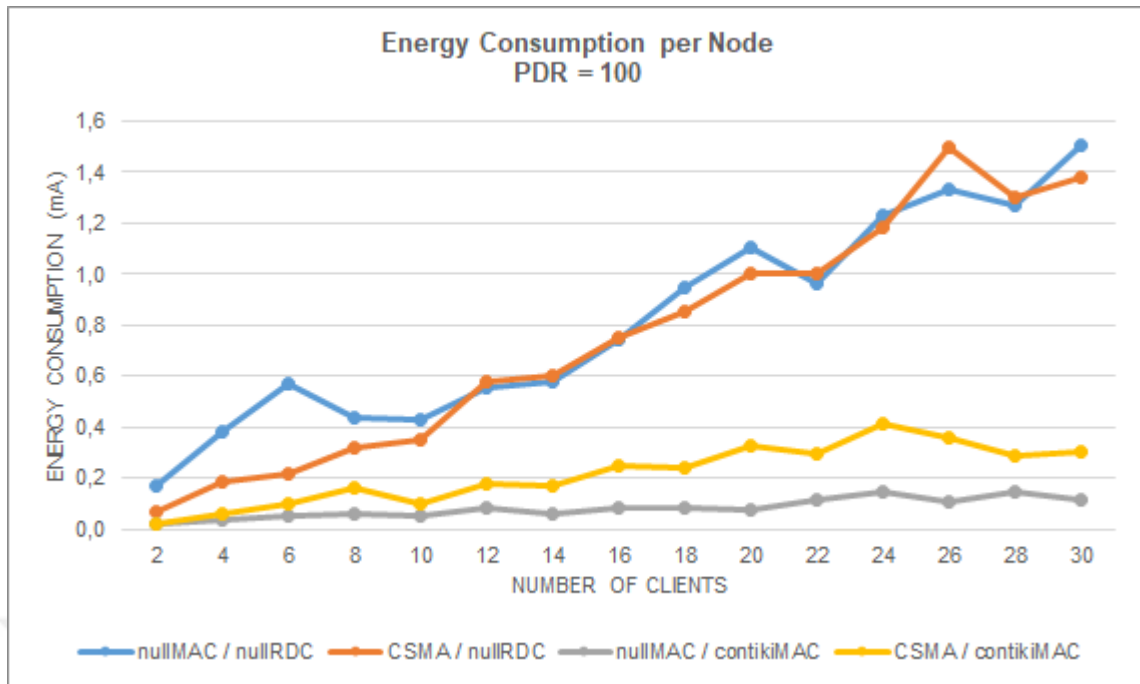


Figure 5.5 - Average Energy Consumption per node for PDR=100

According to the energy consumption data obtained when the PDR value is 100 (Figure 5.5), the energy consumption per node increases in parallel with the increase in the number of nodes. The change in the RDC layer protocol, which plays an active role in energy consumption, is evident in these results. When contikiMAC is used in the RDC layer, there is a clear difference in energy consumption. With the contikiMAC, it was observed that the use of the CSMA protocol in the MAC layer increased energy consumption. However, when nullRDC is used in the RDC layer, the use of CSMA in the MAC layer can reduce energy consumption slightly.

When the PDR value is 95 (Figure 5.6), it is seen that there is an increase in energy consumption in the protocol configuration using nullRDC protocol in RDC layer and nullMAC protocol in MAC layer. The reason for this energy consumption increase might be re-transmissions for packets that cannot be transmitted as a result of a decrease in PDR value.

The energy consumption results in the case that the PDR value is 90 (Figure 5.7) is close to that of the PDR value 95. The difference between Figure 5.6 and Figure 5.7, is not close to the difference between Figure 5.5 and Figure 5.6.

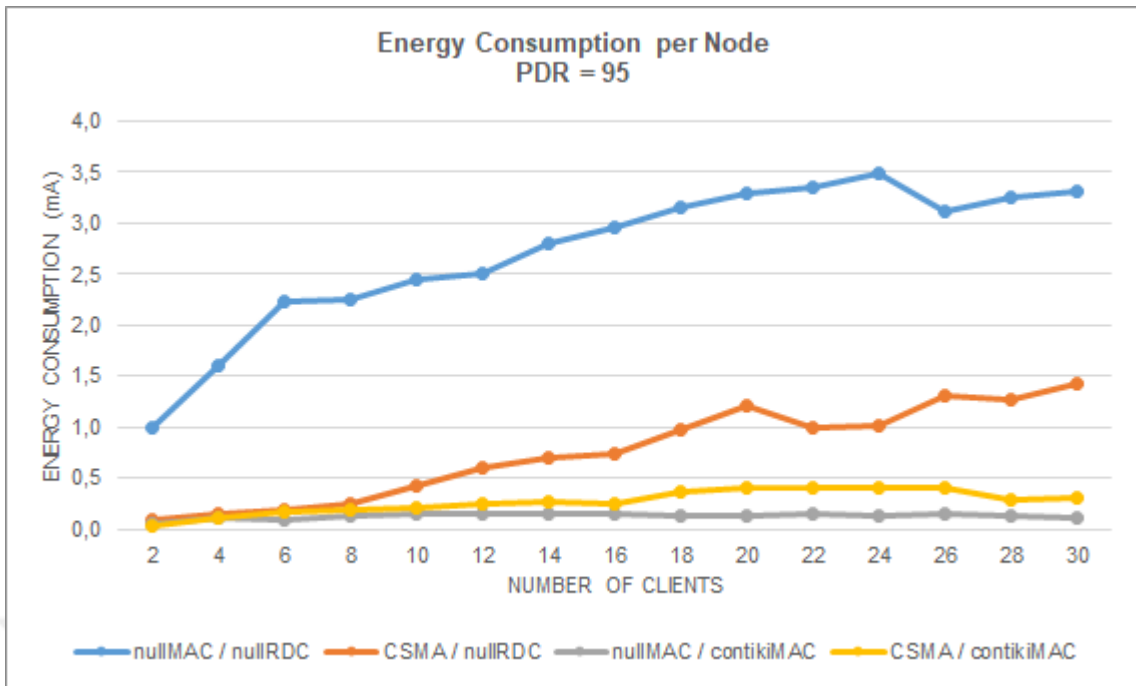


Figure 5.6 - Average Energy Consumption per node for PDR=95

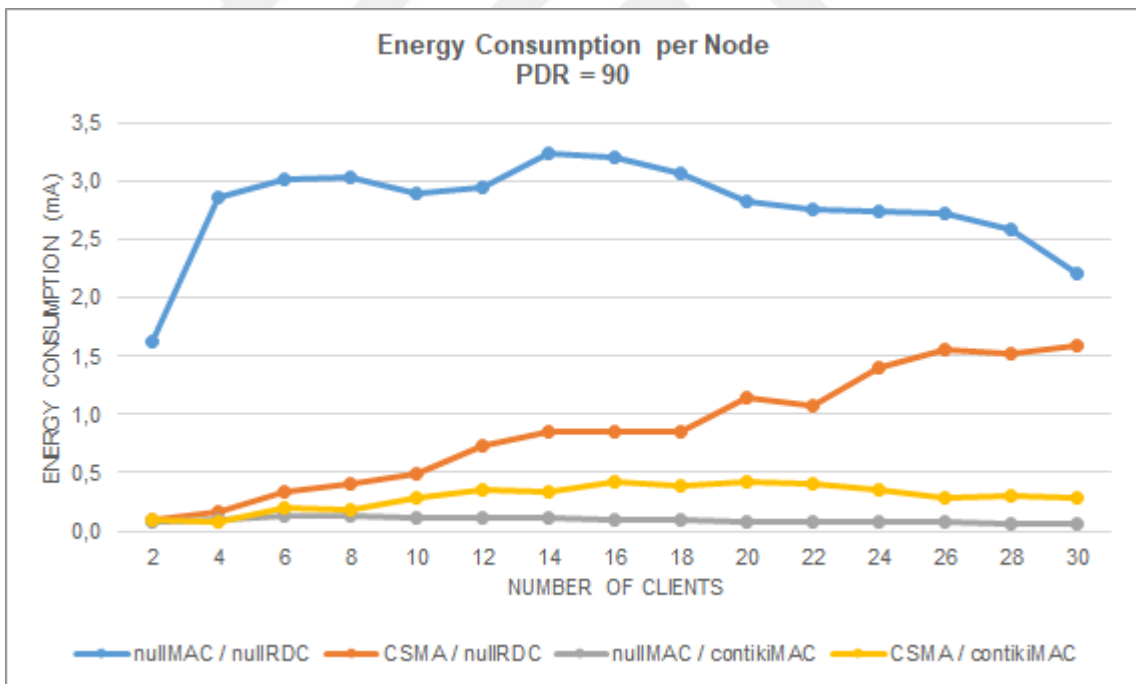


Figure 5.7 - Average Energy Consumption per node for PDR=90

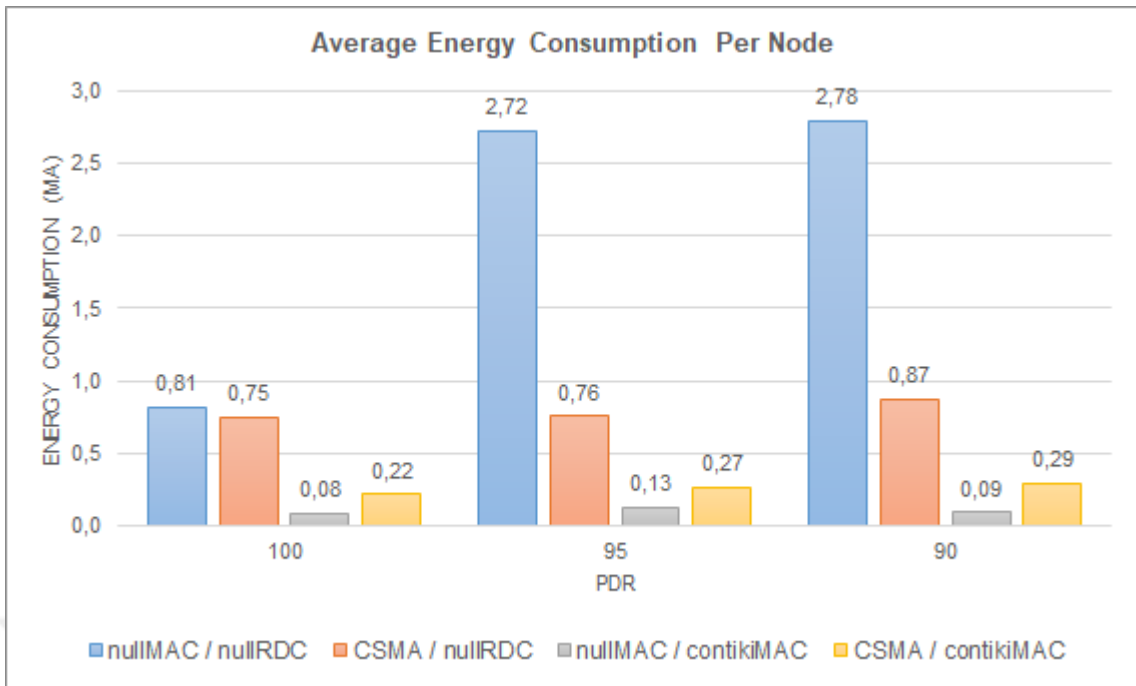


Figure 5.8 - Average Energy Consumption per node for All Network Stacks

When the results are examined, it is clear that the factor that affects the most energy consumption is the RDC protocol. When the RDC protocol is not used, energy consumption is higher. However, the only factor affecting energy consumption is not the RDC protocol. The MAC layer protocol or different PDR values also cause changes in energy consumption. In addition, the increase in the number of nodes usually leads to an increase in energy consumption.

In order to give these results a meaning closer to real life usage, it was calculated how long the 802.15.4 radio can be operated with a battery. Only the 802.15.4 radio is thought to be powered by a relatively small and lightweight CR2032 battery. The energy consumption of health sensors and microcontrollers is ignored. The average capacity of the CR2032 battery is 220 mA. According to this value, it was calculated how many days the 802.15.4 radio could be running. The data are given in Table 5.4.

In addition, the average battery life of each node was calculated for the respective scenario in the bottom line of the table. According to these averages, the graph for battery life for different scenarios and PDR values is given in Figure 5.9.

BATTERY LIFE (Days)												
PDR / Clients	nullMAC / nullRDC			CSMA / nullRDC			nullMAC / contikiMAC			CSMA / contikiMAC		
	100	95	90	100	95	90	100	95	90	100	95	90
2	0,38	0,38	0,38	0,38	0,38	0,38	7,56	7,22	8,31	6,69	6,03	5,39
4	0,38	0,38	0,38	0,38	0,38	0,38	8,04	8,16	8,45	5,93	5,50	5,00
6	0,38	0,38	0,38	0,38	0,38	0,38	7,61	8,20	8,25	5,19	4,99	4,74
8	0,38	0,38	0,38	0,38	0,38	0,38	8,23	7,88	8,08	4,64	5,33	5,67
10	0,38	0,38	0,38	0,38	0,38	0,38	8,04	7,80	8,15	6,14	5,68	5,60
12	0,38	0,38	0,38	0,38	0,38	0,38	6,25	7,58	7,96	4,97	5,15	5,21
14	0,38	0,38	0,38	0,38	0,38	0,38	7,40	7,46	7,99	5,58	5,21	5,21
16	0,38	0,38	0,38	0,38	0,38	0,38	6,88	6,95	7,80	5,11	5,08	4,36
18	0,38	0,38	0,38	0,38	0,38	0,38	6,76	7,15	7,81	4,78	4,65	4,55
20	0,38	0,38	0,38	0,38	0,38	0,38	7,14	6,96	7,78	4,29	4,26	4,15
22	0,38	0,38	0,38	0,38	0,38	0,38	5,61	6,86	8,04	4,28	4,20	4,05
24	0,38	0,38	0,38	0,38	0,38	0,38	5,50	6,72	7,73	4,03	4,15	4,15
26	0,38	0,38	0,38	0,38	0,38	0,38	5,92	6,69	7,69	3,69	3,78	4,07
28	0,38	0,38	0,38	0,38	0,38	0,38	5,84	6,46	7,67	3,99	3,95	3,87
30	0,38	0,38	0,38	0,38	0,38	0,38	6,70	6,38	7,67	3,70	3,79	3,79
AVG	0,4	0,4	0,4	0,4	0,4	0,4	6,9	7,2	8,0	4,9	4,8	4,7

Table 5.4 - Battery Life per Node (in Days)

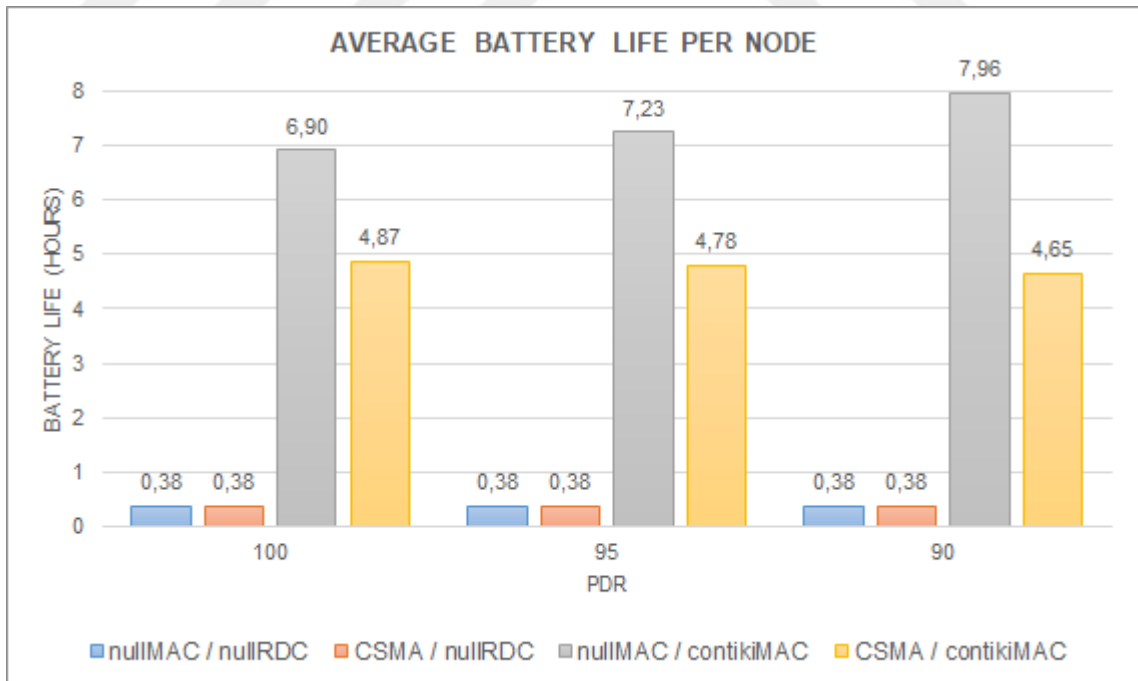


Figure 5.9 - Average Battery Life of Nodes in Different Network Stacks

When no RDC protocol is used, the radio will remain on continuously. As a result, no energy savings will be made, the 802.15.4 radio will always operate at full power. In all

scenarios where the RDC protocol is not used, the energy consumption is constant since the radio is not turned off at all. As can be seen from the table and graph, in these scenarios where RDC is not used, the energy consumption will be high and the battery life will be short. In these cases, it is necessary to change the battery approximately 2 times a day.

When the scenarios using contikiMAC RDC protocol are examined, there is a serious improvement in the battery life. Against the scenarios where the RDC protocol is not used, when the contikiMAC RDC protocol is used, the frequency of battery change is approximately 4.5 - 8 days. Thus, the patient's health data can be observed for a long time with a single battery.

5.3. Reliability

Another metric that affects performance while collecting health data from patients is reliability. Network packets that contain health data from patients can be lost in this lossy network environment. Reliability is a metric that expresses how low these packet losses are. Naturally, low packet loss would mean high reliability. In addition, packet losses delay the acquisition of accurate data. The possibility of problems in the environment increases with the increase of packet losses. Low packet losses usually provide more stable data transfer.

Packet success rates obtained according to our simulation results are as in Table 5.5. As in previous tables, scenarios with one or more nodes unable to send packets for 60 minutes are expressed in gray and underlined.

SUCCESSFUL PACKETS												
PDR / Client	nullMAC / nullRDC			CSMA / nullRDC			nullMAC / contikiMAC			CSMA / contikiMAC		
	100	95	90	100	95	90	100	95	90	100	95	90
2	100,0	100,0	99,5	100,0	100,0	100,0	100,0	99,5	98,0	100,0	99,5	100,0
4	100,0	94,8	85,8	100,0	100,0	100,0	99,0	93,5	65,8	99,8	99,0	99,0
6	99,7	92,7	76,5	99,8	99,8	100,0	98,0	80,2	66,2	97,7	98,0	95,3
8	99,8	94,5	71,6	100,0	99,8	99,9	97,3	87,0	62,0	96,8	95,3	94,3
10	99,3	89,2	73,0	99,6	99,9	99,8	95,3	75,0	53,7	93,6	92,0	87,7
12	99,0	92,3	67,3	99,9	99,9	99,9	94,6	74,6	50,9	90,8	91,6	86,1
14	98,8	87,5	67,9	99,8	99,8	99,7	93,1	69,9	46,0	85,6	86,6	80,0
16	98,9	88,5	63,6	99,4	99,6	99,6	88,6	65,5	38,6	76,3	71,2	69,8
18	98,2	83,6	60,8	99,4	99,7	98,8	86,2	56,5	36,7	75,4	70,4	63,8
20	97,1	84,8	56,7	99,1	99,4	99,5	85,5	52,6	29,4	65,8	53,2	54,1
22	97,0	81,0	53,5	99,0	99,0	98,8	82,7	52,6	27,5	54,7	52,9	40,9
24	97,3	78,0	50,8	98,8	98,5	98,4	76,0	45,1	25,5	41,3	48,0	37,8
26	92,4	76,3	48,8	98,3	98,1	98,5	75,0	44,0	26,8	44,5	37,7	26,8
28	95,3	75,1	46,4	97,9	97,8	97,6	70,5	38,5	24,3	28,9	29,1	27,3
30	95,1	69,5	41,2	97,3	96,5	97,5	70,3	34,3	23,7	33,0	27,8	21,0
AVG	97,8	85,9	64,2	99,2	99,2	99,2	87,5	64,6	45,0	72,3	70,1	65,6

Table 5.5 – Average Successful Packets

As can be seen from the table, the number of successful packets is generally high in network environments with a low number of clients. As the number of nodes on the network increases, the rate of successful packet decreases due to increased network traffic. Different graphics are obtained according to different PDR values in order to show the data in the table more clearly. Charts with PDR values of 100, 95 and 90 are given in Figure 5.10, Figure 5.11 and Figure 5.12, respectively. Apart from these graphs, a graph comparing the average number of successful packets of different scenarios is given in Figure 5.13.

The result graph with a PDR of 100 is given in Figure 5.10. When the graph is examined, the differences of different protocol configurations for reliability metrics can be seen. According to these results, the configuration that provides the best values is the configuration using the null layer in the MAC layer with the CSMA protocol and the RDC layer. However, when using contikiMAC in the RDC layer, selecting the CSMA protocol in the MAC layer makes the results worse. In cases where the number of nodes is low, the reliability result values are good, while the increase in the number of nodes has reduced the reliability values, especially when contikiMAC protocol is used in the RDC layer.

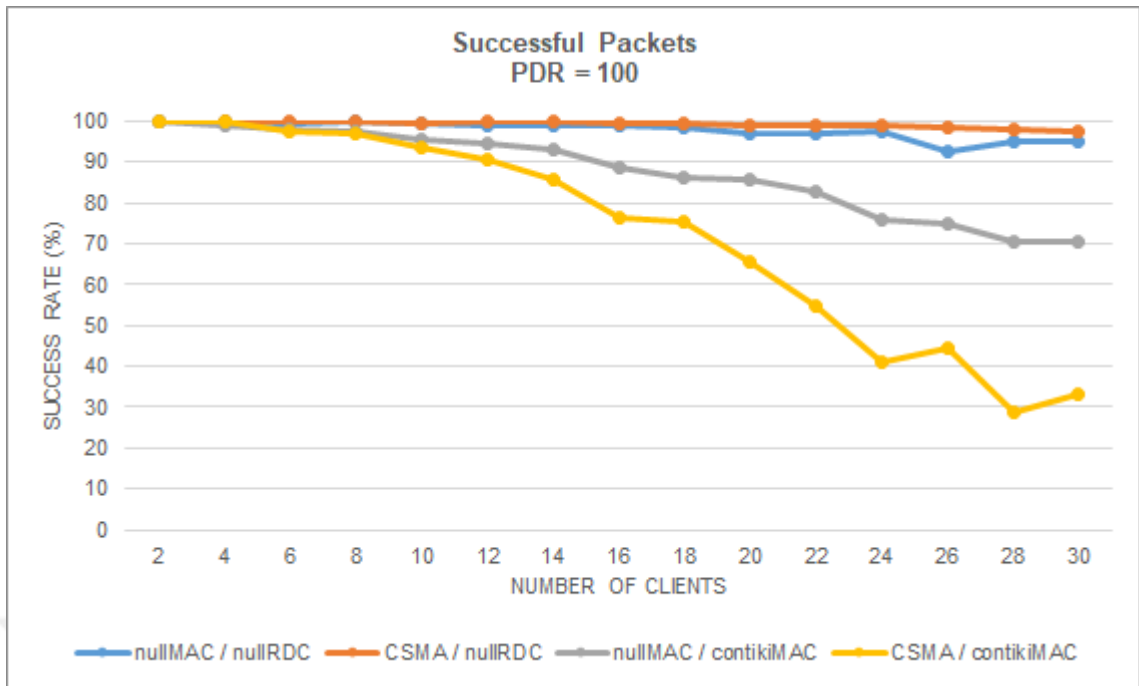


Figure 5.10 – Successful Packet Ratio for PDR = 100

The results of CSMA / nullRDC and CSMA / contikiMAC protocol configurations are not affected much when the PDR value is reduced to 95 in Figure 5.11. However, there has been a significant decrease in the reliability metric values for the nullMAC / nullRDC and nullMAC / contikiMAC protocol configurations. In particular, the use of the CSMA protocol in the MAC layer can be said to help maintain reliability values against decreases in PDR values.

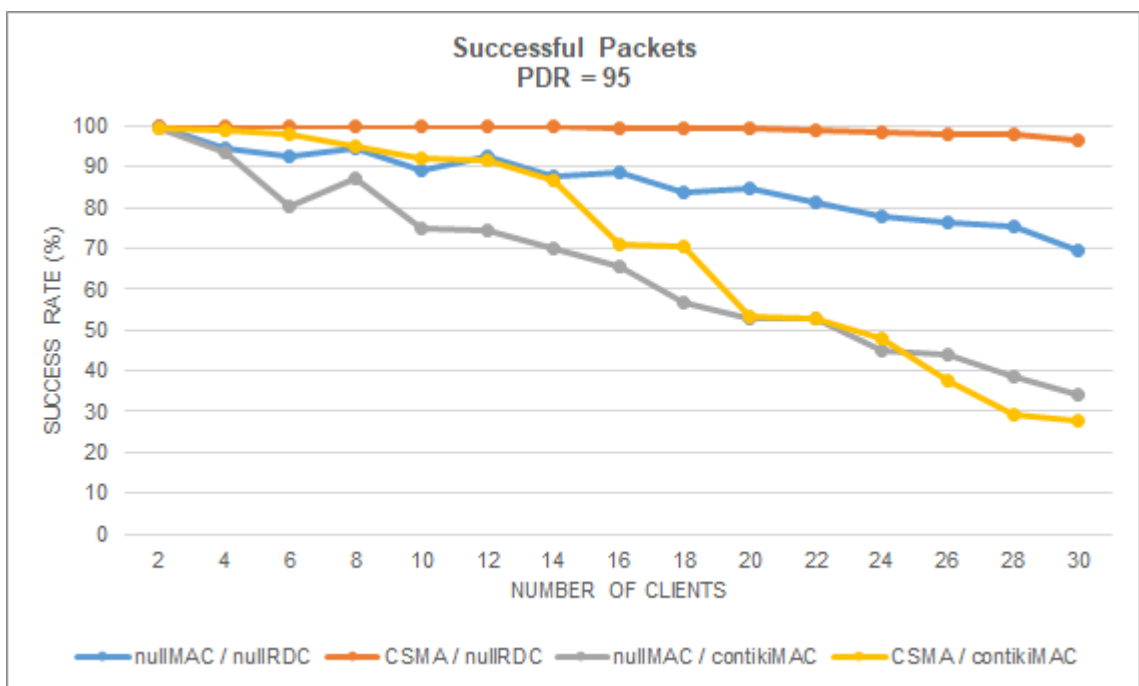


Figure 5.11 – Successful Packet Ratio for PDR = 95

When the PDR values are slightly lowered to 90 in Figure 5.12, it can be seen that the reliability values of the nullMAC / nullRDC and nullMAC / contikiMAC configurations are even lower. When we look at the CSMA / nullRDC and CSMA / contikiMAC configurations, there is little difference in reliability values.

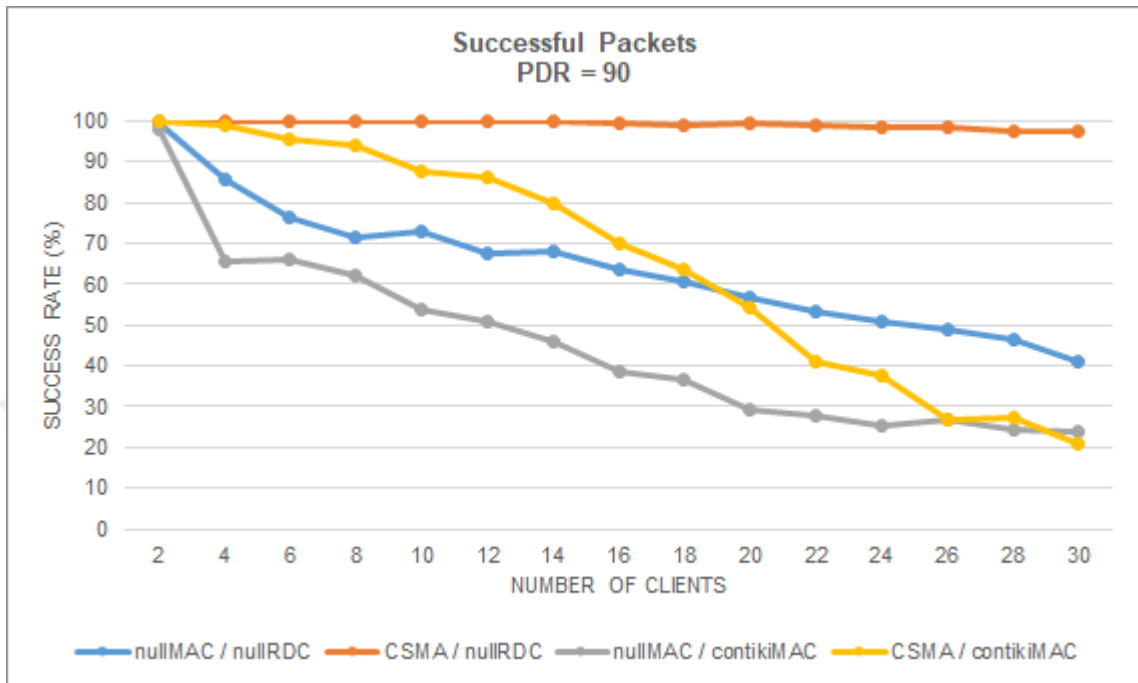


Figure 5.12 – Successful Packet Ratio for PDR = 90

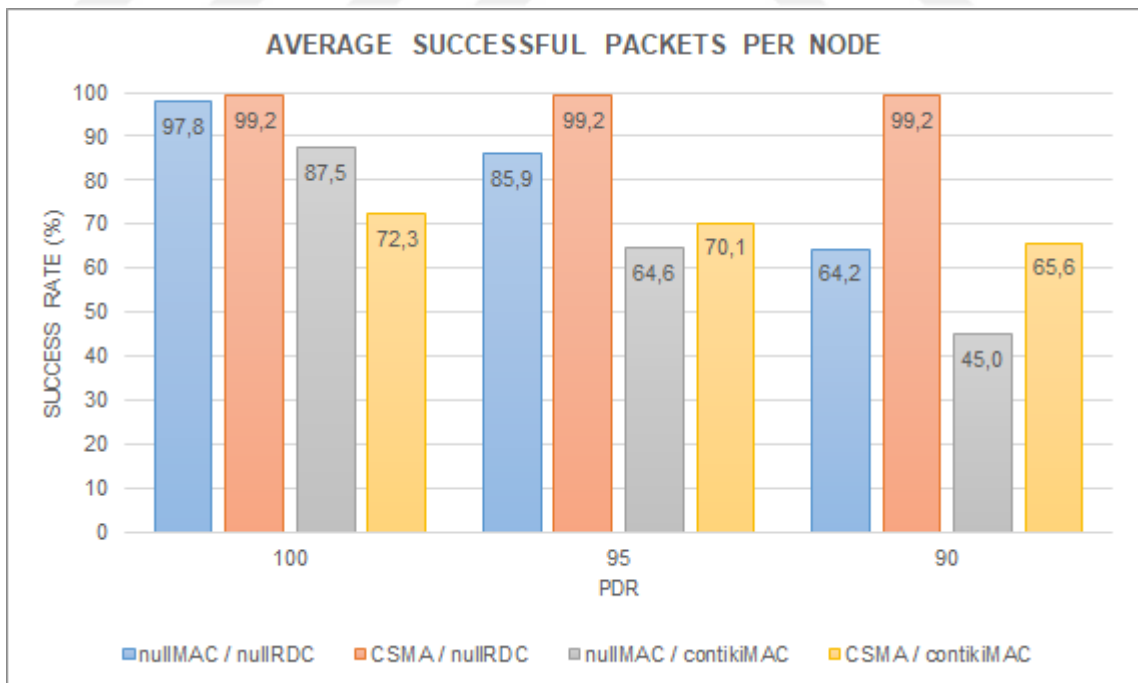


Figure 5.13 – Average Successful Packet Ratio for Different Network Stacks

When we look at Figure 5.13, which shows the overall average reliability values, it can be observed how the reliability values change according to the PDR values. It is clear

that the reliability values of the nullMAC / nullRDC and nullMAC / contikiMAC configurations are significantly reduced. In this graph, it can be seen that using the CSMA protocol in the MAC layer helps to keep reliability values stable.

5.4. Throughput

Throughput is another metric used in performance assessments in the transmission of health data. With the throughput metric, it can be determined what the packet transmission speed of a node will be. The faster a packet can send, the more time it will be reserved for other nodes to communicate. In addition, rapid packet transmission will allow for more health data to be transmitted within a given time frame.

In this study, only successful packets are considered when calculating the throughput metric. Lost packets, ack and signal packets are not included in the calculation. Throughput can be analyzed in two different ways: by node or by network. Both analyses were conducted in this study.

As in previous metrics, the conditions that are invalid for obtaining health data are underlined and colored in gray.

5.4.1. Node Throughput

When examining the throughput metric for the node, the bandwidths of the nodes are analyzed. These results are obtained by calculating the total of the successful packets each node sends within one second.

The data of the throughput based on the results obtained are given in Table 5.6. The data in this table only covers packets that have been correctly transmitted. Lost packets, ack and signal packets are not included. In the table, each node's throughput values are given for different protocol stacks and different counts of nodes. Average values for different protocol stacks are also given in the bottom line. In order to increase the visibility of the data in the table, graphs based on the data in the table were obtained. Figure 5.14, Figure 5.15, and Figure 5.16 show the throughput values for PDR values 100,95 and 90 in different network stacks, respectively.

NODE THROUGHPUT (PPS)												
PDR / Client	nullMAC / nullRDC			CSMA / nullRDC			nullMAC / contikiMAC			CSMA / contikiMAC		
	100	95	90	100	95	90	100	95	90	100	95	90
2	3,86	0,67	0,41	10,46	8,15	7,53	1,93	0,80	0,42	1,62	1,46	0,50
4	1,77	0,39	0,20	3,68	4,45	4,02	0,90	0,27	0,21	0,75	0,45	0,71
6	1,16	0,28	0,17	3,09	3,74	1,99	0,65	0,27	0,16	0,49	0,30	0,25
8	1,53	0,28	0,16	2,09	2,68	1,64	0,48	0,22	0,15	0,33	0,24	0,24
10	1,54	0,24	0,17	1,90	1,57	1,37	0,62	0,16	0,14	0,40	0,19	0,14
12	1,19	0,25	0,15	1,16	1,11	0,92	0,46	0,17	0,14	0,26	0,19	0,12
14	1,14	0,21	0,14	1,11	0,95	0,79	0,56	0,15	0,12	0,23	0,15	0,12
16	0,89	0,20	0,13	0,89	0,90	0,78	0,38	0,16	0,12	0,15	0,14	0,10
18	0,69	0,18	0,13	0,78	0,68	0,77	0,40	0,15	0,13	0,17	0,10	0,09
20	0,59	0,17	0,13	0,66	0,55	0,58	0,40	0,14	0,13	0,12	0,08	0,08
22	0,67	0,16	0,13	0,66	0,66	0,61	0,32	0,13	0,12	0,11	0,08	0,06
24	0,53	0,15	0,12	0,56	0,65	0,47	0,24	0,14	0,12	0,06	0,07	0,07
26	0,46	0,16	0,12	0,44	0,50	0,42	0,29	0,12	0,12	0,09	0,06	0,06
28	0,50	0,15	0,12	0,50	0,51	0,43	0,21	0,12	0,12	0,06	0,07	0,06
30	0,42	0,14	0,12	0,47	0,45	0,41	0,23	0,12	0,12	0,07	0,06	0,05
AVG	1,13	0,24	0,16	1,90	1,84	1,52	0,54	0,21	0,16	0,33	0,24	0,18

Table 5.6 - Node Throughput (PPS-Packet per Second)

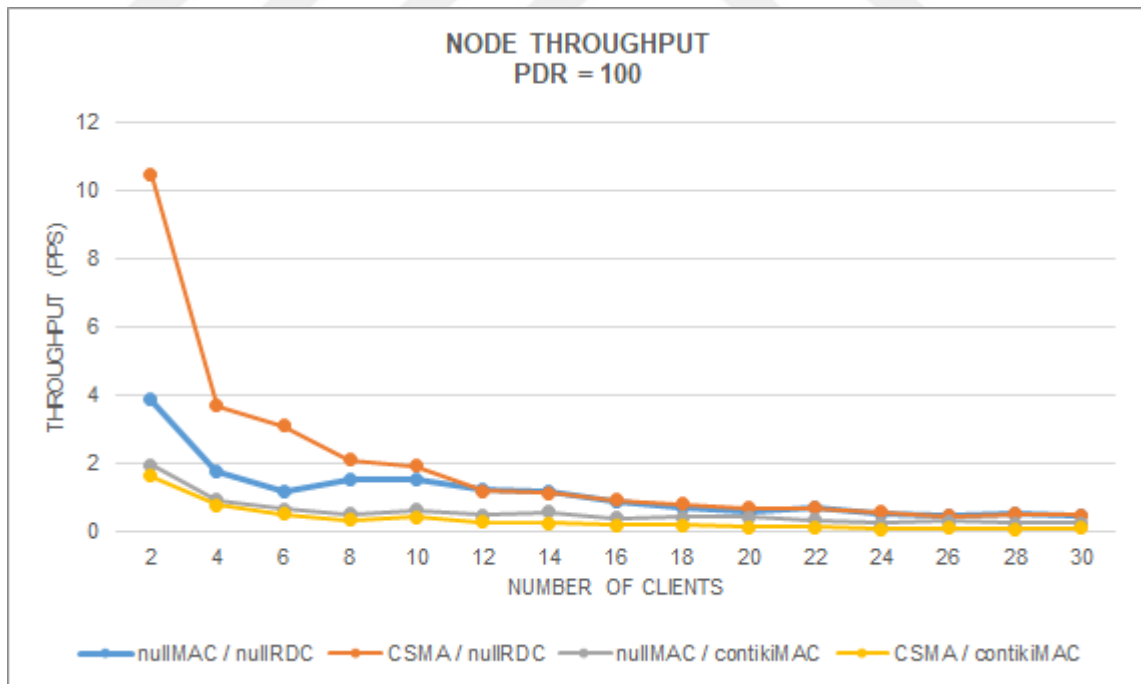


Figure 5.14 - Node Throughput for PDR = 100

Figure 5.14 shows node throughput of different protocol configurations with PDR value of 100. As can be seen, the best values for all protocol configurations are obtained with the minimum number of nodes. By increasing the number of nodes, the throughput

value per node is reduced. What is noteworthy here is that the CSMA / nullRDC protocol configuration often achieves better throughput values. The nullMAC / nullRDC protocol configuration has almost the same throughput values as the CSMA / nullRDC protocol configuration when the number of nodes increases. Hem nullMAC / contikiMAC hem de CSMA / contikiMAC protokol konfigürasyonları daha düşük çıkış değerleri gösterir.

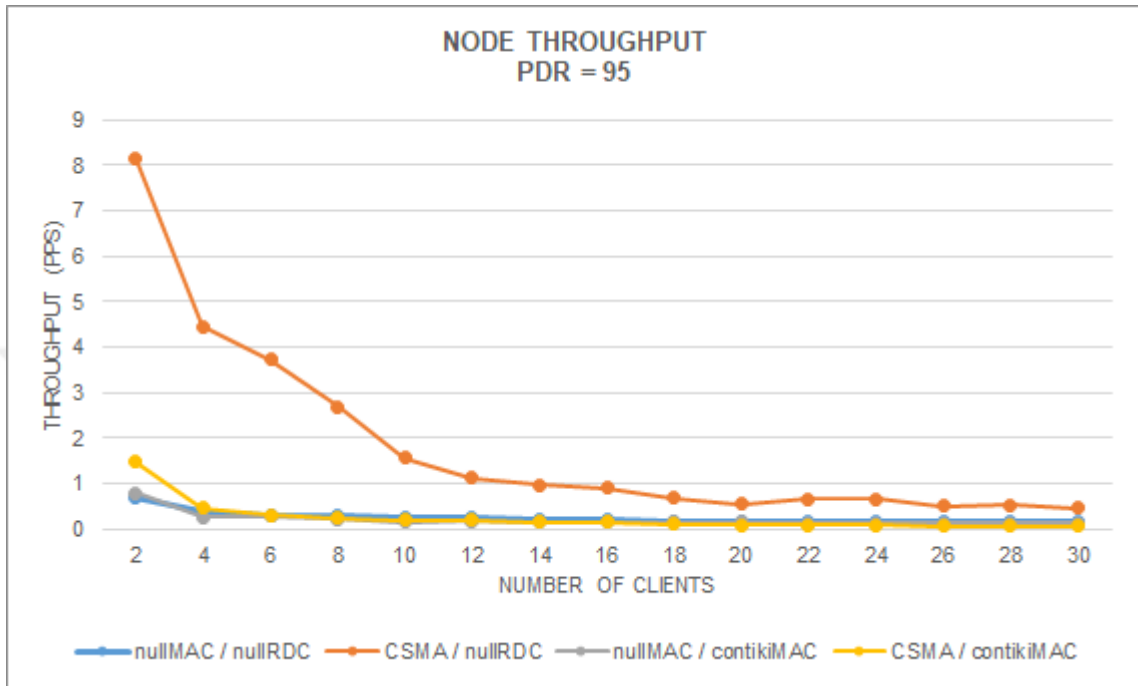


Figure 5.15 - Node Throughput for PDR = 95

Where PDR values are 95 (Figure 5.15) and 90 (Figure 5.16), they have similar graphs with each other. For both cases, the CSMA / nullRDC protocol configuration achieves the best throughput values. The other three protocol configurations (nullMAC / nullRDC, nullMAC / contikiMAC, CSMA / contikiMAC) have reached very similar throughput values. Although the CSMA / contikiMAC protocol configuration has slightly higher values in cases with low number of nodes, with the increase of the node, this difference is lost. For most cases with PDR values lower than 100, throughput value is lower than one packet per second.

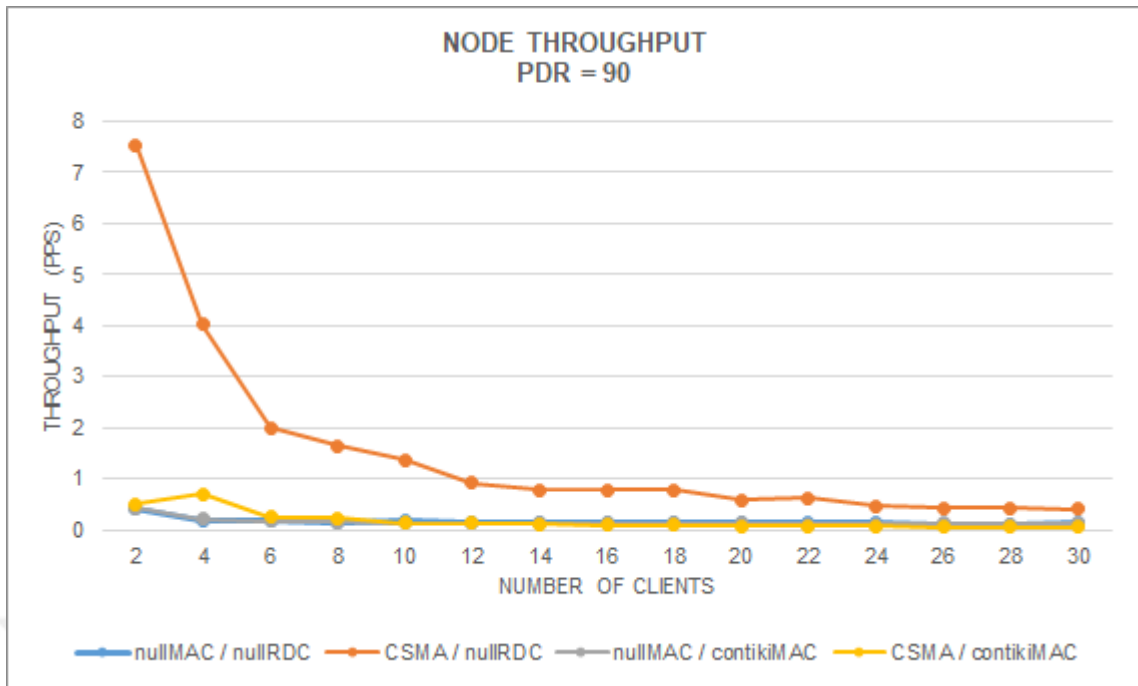


Figure 5.16 - Node Throughput for PDR = 90

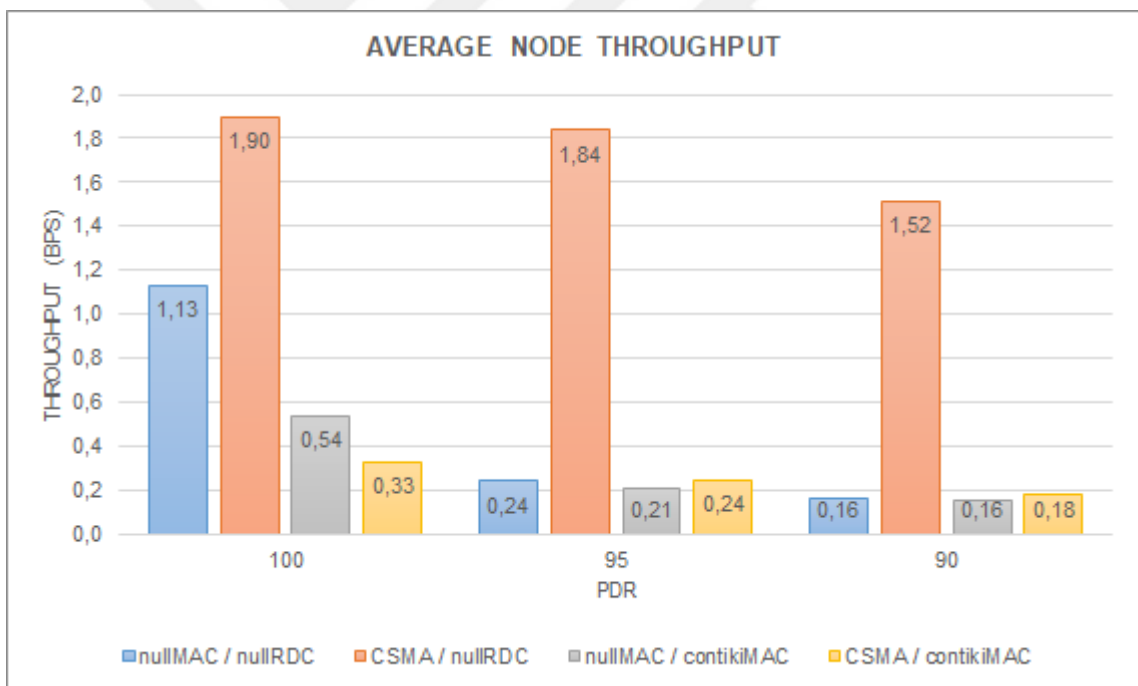


Figure 5.17 - Average Node Throughput for Different Network Stacks

According to the data obtained, the average and summary of the throughput values are given in Figure 5.17. When the results were examined, interestingly, when a protocol was not used in the MAC and RDC layer, the results were effective when the PDR value was 100. However, as the PDR decreases, a significant decrease is observed in throughput values. This is because, as the PDR value decreases, the packet transmission rate decreases. As mentioned earlier, factors such as signal quality and interference can affect

the PDR value in the real environment. It is not always possible to have a PDR of 100 in the real environment. Using the CSMA protocol in the MAC layer and not using the protocol in the RDC layer, all the PDR values had a significant stability. While no protocol was used in the MAC layer, the use of contikiMAC in the RDC layer also negatively affected the throughput performance. The use of the CSMA protocol in the MAC layer with the contikiMAC protocol in the RDC layer was able to improve the results slightly.

5.4.2. NetworkThroughput

As well as the amount of throughput of the nodes, the total throughput on the network is also an important metric for performance measurement. In this way, the maximum number of packets that can be sent on the network in unit time can be determined. In the light of this data, it can be deduced how successful the network may be when a different data packet is in question. With the simulations, the network's throughput capacity was determined under different scenarios. The results obtained are shown in Table 5.7 as a table, and the graphs obtained from the same data are given in Figure 5.18, Figure 5.19 and Figure 5.20, with PDR values 100, 95 and 90 respectively.

NETWORK THROUGHPUT (PPS)												
PDR / Client	nullMAC / nullRDC			CSMA / nullRDC			nullMAC / contikiMAC			CSMA / contikiMAC		
	100	95	90	100	95	90	100	95	90	100	95	90
2	7,72	1,34	0,82	20,92	16,30	15,06	3,86	1,60	0,85	3,25	2,92	1,01
4	7,06	1,58	0,80	14,73	17,81	16,08	3,60	1,06	0,83	3,01	1,79	2,83
6	6,99	1,66	1,01	18,52	22,42	11,95	3,88	1,63	0,97	2,95	1,82	1,49
8	12,22	2,24	1,26	16,73	21,43	13,10	3,86	1,78	1,20	2,62	1,93	1,89
10	15,36	2,43	1,69	19,00	15,68	13,74	6,23	1,58	1,39	4,03	1,92	1,39
12	14,33	2,94	1,83	13,91	13,36	11,01	5,47	2,06	1,63	3,10	2,22	1,46
14	15,95	2,92	1,96	15,47	13,35	11,00	7,80	2,14	1,71	3,27	2,15	1,65
16	14,17	3,19	2,13	14,23	14,35	12,53	6,03	2,48	1,99	2,48	2,24	1,54
18	12,44	3,20	2,39	13,98	12,26	13,94	7,21	2,70	2,27	3,03	1,88	1,68
20	11,73	3,45	2,67	13,22	10,98	11,55	7,96	2,86	2,65	2,41	1,57	1,54
22	14,85	3,55	2,85	14,48	14,51	13,52	7,09	2,96	2,63	2,45	1,77	1,40
24	12,72	3,58	2,98	13,42	15,52	11,29	5,68	3,26	2,87	1,53	1,76	1,58
26	12,09	4,26	3,10	11,38	12,96	11,01	7,60	3,09	3,24	2,21	1,63	1,52
28	14,02	4,33	3,37	14,07	14,34	11,99	6,00	3,38	3,25	1,79	1,84	1,69
30	12,67	4,21	3,73	14,16	13,57	12,25	6,86	3,75	3,66	2,22	1,88	1,50
AVG	12,29	2,99	2,17	15,22	15,26	12,67	5,94	2,42	2,08	2,69	1,96	1,61

Table 5.7 - Network Throughput (PPS-Packet per Second)

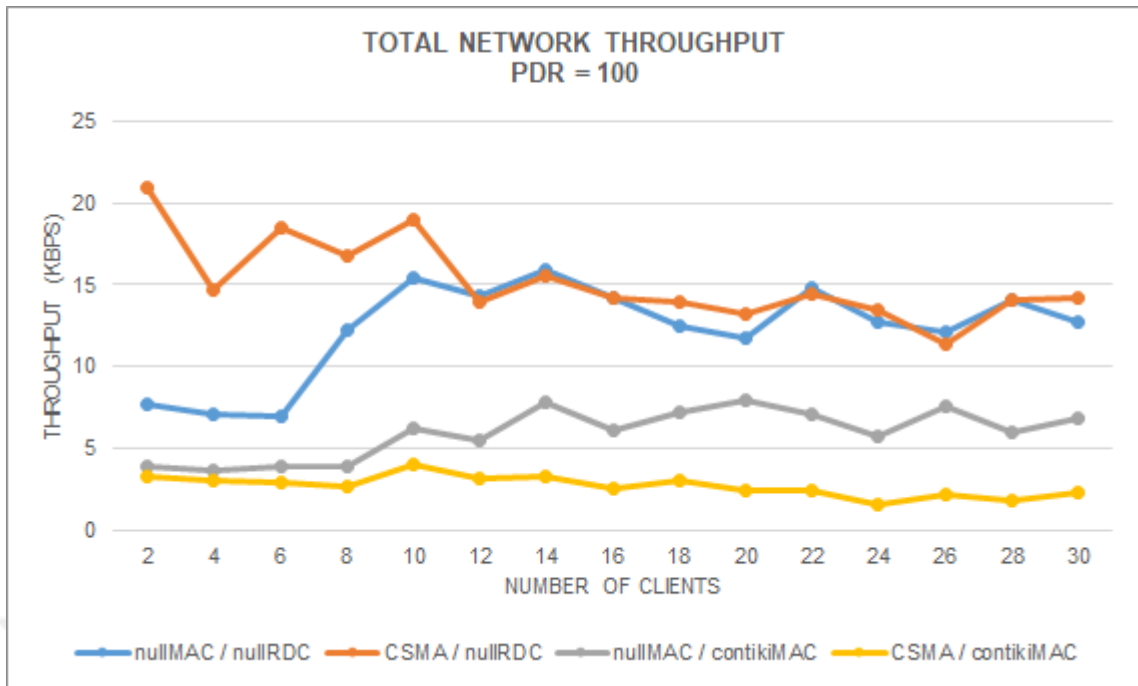


Figure 5.18 - Network Throughput for PDR = 100

When we examine the network throughput value in cases where the PDR is 100 (Figure 5.18), we can see that the CSMA / nullRDC protocol configuration gives higher values in the low number of nodes. The nullMAC / nullRDC configuration gives low throughput values in low node counts according to the CSMA / nullRDC configuration. The increase in the number of nodes allowed a slight increase in the throughput values for this configuration. The nullMAC / contikiMAC and CSMA / contikiMAC configurations have similar throughput values where the number of nodes is small. As a result of the increase in the number of nodes, although the throughput value of the nullMAC / contikiMAC configuration has increased, the same cannot be said for the CSMA / contikiMAC configuration.

When we examine the situation where the PDR value is 95 (Figure 5.19), we can observe that the CSMA / nullRDC configuration continues with similar throughput values with the PDR value of 100. There is a significant decrease in the throughput values of the nullMAC / nullRDC configuration. The nullMAC / contikiMAC configuration also reduced the throughput value to less than 5 packets per second. Although the CSMA / contikiMAC configuration has experienced a decrease in throughput values, this is not a significant reduction.

When the PDR value is reduced to 90, there is a significant decrease in the average of the throughput values. The change in throughput values with respect to the increasing number of nodes is similar to that in which the PDR value is 95.

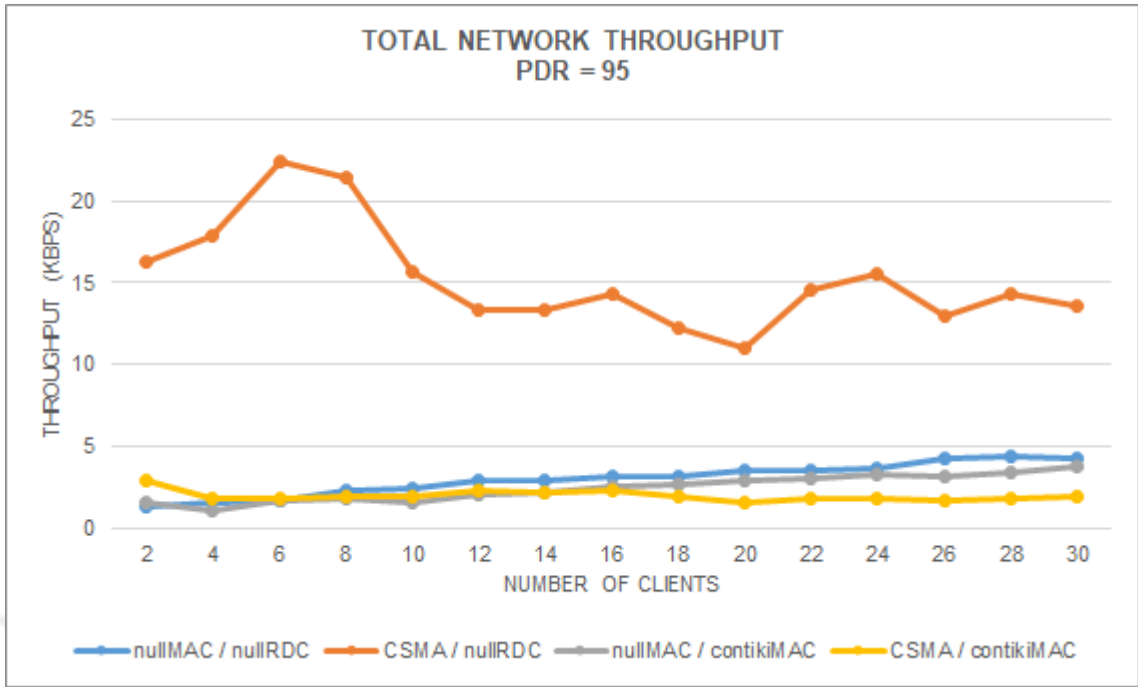


Figure 5.19 - Network Throughput for PDR = 95

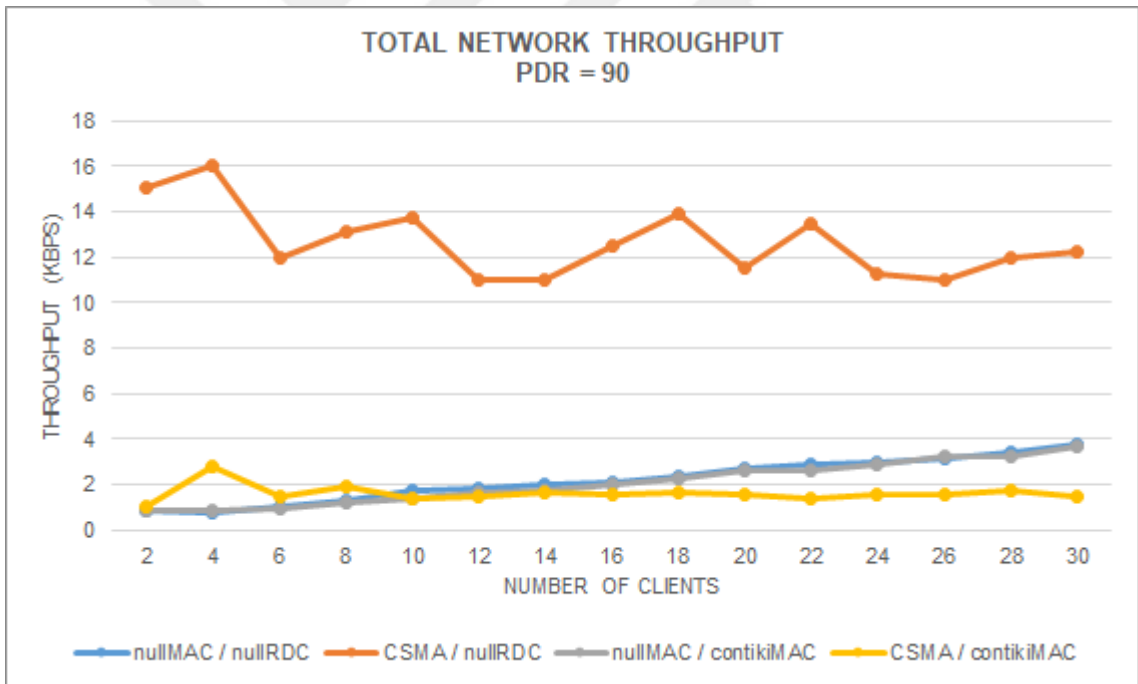


Figure 5.20 - Network Throughput for PDR = 90

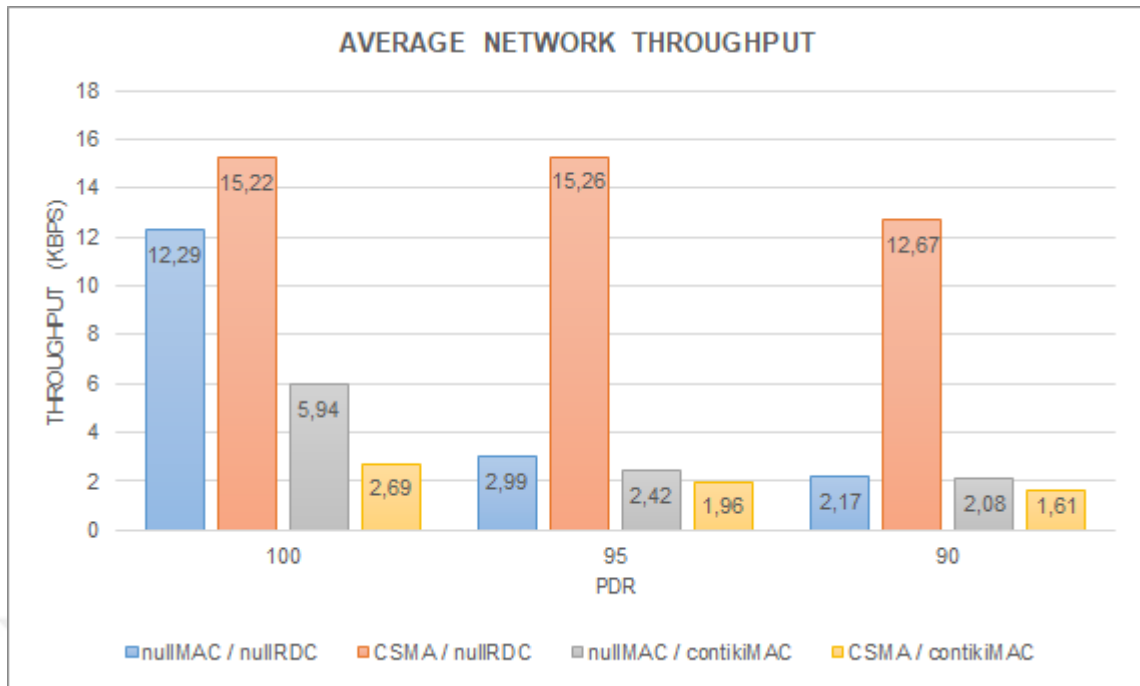


Figure 5.21 - Average Network Throughput for Different Protocol Stacks

Figure 5.21 shows the average of all network throughput data. The change in network throughput values for different protocol stacks can be seen more clearly with this graph. It was observed that the changes in the protocols in the RDC and MAC layers affected the performance in these results, which are similar to the nodes throughput results. According to the numerical values of these results, the packet sending capacity of the network has emerged.

5.5. Selecting the Valid Protocol Stacks

Performance metrics are used to determine the protocol stacks that may be valid in the transmission of health data. The most important criterion for the transmission of health data is the transmission of the obtained health data to the health center without losing its validity. In this context, the most important metric will be the latency and reliability metric. As stated in the previous sections, the importance of health data is different according to different patient groups. In this case, the stack of protocols to be selected may also vary according to the patient and the type of health data. In addition, energy efficiency, which is another metric, can be effective in the selection of protocols. As an example, there is no mobility for a patient in continuous sleep. For a patient in this condition, energy efficiency can be ignored. In return, better results can be achieved in the latency metric and other metrics. Based on such scenarios, ideal protocol stacks can be determined by patient groups and sensor types. When making these selections, metrics should be considered in order of importance. In the following section, these ideal protocol stacks were examined.

5.6. Determination of Optimal IoT Stack to Transfer Health Data

When all the results are examined, different protocol stacks can be suggested according to patient groups and sensor types. There is no best stack of protocols available for all patients. Instead, we can offer the best protocol stacks according to patient groups and sensor types. It should also be noted that not all sensors mentioned in this study will be present on the patient at the same time. Depending on the type of sensor to be found on the patient, such adjustments can be made easily.

The number of nodes (patients) in the environment is another important factor affecting performance. As with other factors affecting performance, the number of patients also affects the protocol stack to be selected. In scenarios with a large number of patients, it may not be possible to obtain data from certain sensor types at valid times. Considering such cases, tables prepared according to patient groups and sensor types, indicating the most appropriate protocol stack, are given in Table 5.8 and Table 5.9. In Table 5.8, cases where the number of patients were up to 15 were taken, while in Table 5.9, the number of patients was between 15-30.

Up to 15 nodes in Network			
Sensor	Patient Category		
	Critical	Non-Critical	Follow-up
Body Position	CSMA / contikiMAC	CSMA / contikiMAC	CSMA / contikiMAC
Body Temperature	CSMA / contikiMAC	CSMA / contikiMAC	CSMA / contikiMAC
Snore Sensor	CSMA / contikiMAC	CSMA / contikiMAC	CSMA / contikiMAC
Galvanic Skin Response	CSMA / contikiMAC	CSMA / contikiMAC	CSMA / contikiMAC
Airflow Sensor	CSMA / contikiMAC	CSMA / contikiMAC	CSMA / contikiMAC
EMG Sensor	CSMA / nullRDC	CSMA / contikiMAC	CSMA / contikiMAC
ECG Sensor	CSMA / nullRDC	CSMA / contikiMAC	CSMA / contikiMAC
SPo2	CSMA / nullRDC	CSMA / contikiMAC	CSMA / contikiMAC
Blood Pressure Sensor	CSMA / contikiMAC	CSMA / contikiMAC	CSMA / contikiMAC
Body Scale	CSMA / contikiMAC	CSMA / contikiMAC	CSMA / contikiMAC
Glucometer	CSMA / contikiMAC	CSMA / contikiMAC	CSMA / contikiMAC
Patient ID	-	-	-
Thresold 1: 1 minute		Thresold 2: 15 minutes	
Thresold 3: 30 minutes		Thresold 4: 60 minutes	

Table 5.8 - Ideal Protocol Stack for Different Groups (for up to 15 nodes)

15 to 30 nodes in Network			
Sensor	Patient Category		
	Critical	Non-Critical	Follow-up
Body Position	CSMA / nullRDC	CSMA / nullRDC	CSMA / nullRDC
Body Temperature	CSMA / nullRDC	CSMA / nullRDC	CSMA / contikiMAC
Snore Sensor	CSMA / nullRDC	CSMA / contikiMAC	CSMA / contikiMAC
Galvanic Skin Response	CSMA / contikiMAC	CSMA / contikiMAC	CSMA / contikiMAC
Airflow Sensor	CSMA / nullRDC	CSMA / nullRDC	CSMA / contikiMAC
EMG Sensor	N/A	CSMA / contikiMAC	CSMA / contikiMAC
ECG Sensor	N/A	CSMA / contikiMAC	CSMA / contikiMAC
SPo2	N/A	CSMA / contikiMAC	CSMA / contikiMAC
Blood Pressure Sensor	CSMA / nullRDC	CSMA / contikiMAC	CSMA / contikiMAC
Body Scale	CSMA / contikiMAC	CSMA / contikiMAC	CSMA / contikiMAC
Glucometer	CSMA / contikiMAC	CSMA / contikiMAC	CSMA / contikiMAC
Patient ID	-	-	-
Thresold 1: 1 minute		Thresold 2: 15 minutes	
Thresold 3: 30 minutes		Thresold 4: 60 minutes	

Table 5.9 - Ideal Protocol Stack for Different Groups (for 15-30 nodes)

The collision avoidance feature of the CSMA protocol also prevents delays due to collisions. This makes it possible to obtain the best latency values when using the CSMA protocol. As already mentioned, energy efficiency is an important issue in IoT devices. Therefore, contikiMAC is preferred as RDC protocol whenever possible. For these reasons, priority has been given to the selection of the CSMA protocol in the MAC layer, and the selection of the contikiMAC protocol in the RDC layer.

As can be seen in the table, in a network with a large number of nodes, there may be no suitable stack of protocols for receiving data from specific sensor types. In these scenarios, it would not be reasonable to obtain health data over the IoT network. The scenarios found in this case are marked with N/A in the table.

6. CONCLUSION

In the study on the transmission of health data in an IoT network using 6LoWPAN infrastructure, analysis of different protocol stacks were performed. These performance evaluations were the outcome of the stages performed respectively. Firstly, similar studies in the literature have been researched and the path to be followed in the study has been established. Then, the real hardware environment in which sample health data can be obtained is created. The health data samples obtained in this environment were stored for use in performance evaluations. This step is also a simple proof that health data can be transmitted over the IoT network. In this section, the stages of obtaining health data are detailed. By examining these stages, information can be obtained about how to set up the aforementioned system or similar system. After this stage which was a milestone for the study, the stages involving the performance tests were started.

The next step was to prepare the simulation environment where performance evaluations would be made. In the simulation environment, which was created in close proximity to the actual scenarios, the previously obtained health data was used. Simulation environments that can provide this were investigated and the most appropriate one was chosen. The successful transmission of health data in the simulation environment enabled us to proceed to the next step.

The next step was to examine the change in performance using different protocols. Different protocol stacks have been tried in order to transmit these transmitted health data more effectively and rapidly. Analysis were made using different node (patient) numbers with these protocol changes. According to the analysis, it was investigated which protocol stacks are useful in the transmission of health data. These evaluations were made according to metrics that could make the patients' health data meaningful. Some of the metrics selected at this stage were critical in the transmission of health data, while other metrics were related to how quickly and accurately the health data were sent.

In addition, performance comparisons have been made between protocol stacks available for transmission of health data. According to these performance evaluations, the most appropriate protocol stacks were determined according to patient groups and sensor types to be used. It has been shown that the health data of the patients can be taken without any problem when using these protocol stacks.

As performance analysis of different protocols are given in detail in the study, when a different health data application (sensor) is used, the possible situations can be predicted based on this data. For example, for a different group of patients who require different health

data than the one in this study, the appropriate protocol stack can be proposed by examining the results of the analysis.

In future studies, it is planned to evaluate different protocols in the MAC and RDC layers. In addition, for MAC, RDC or other layer(s), it is planned to develop a protocol which may be more efficient in transferring health data.



7. REFERENCES

- Atzori, Luigi, Antonio Iera, and Giacomo Morabito. 2010. "The Internet of Things: A Survey." *Computer networks* 54(15): 2787–2805.
- Baccelli, Emmanuel et al. 2013. "RIOT OS: Towards an OS for the Internet of Things." In *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 79–80.
- Baker, Stephanie B., Wei Xiang, and Ian Atkinson. 2017. "Internet of Things for Smart Healthcare: Technologies, Challenges, and Opportunities." *IEEE Access* 5: 26521–44.
- Bandyopadhyay, Debasis, and Jaydip Sen. 2011. "Internet of Things: Applications and Challenges in Technology and Standardization." *Wireless Personal Communications* 58(1): 49–69.
- Betzler, August, Carles Gomez, Ilker Demirkol, and Josep Paradells. 2015. "CoCoA+: An Advanced Congestion Control Mechanism for CoAP." *Ad Hoc Networks* 33: 126–39.
- Bormann, Carsten, Angelo P. Castellani, and Zach Shelby. 2012. "CoAP: An Application Protocol for Billions of Tiny Internet Nodes." *IEEE Internet Computing* 16(2): 62–67.
- Chen, Yibo et al. 2011. "6LoWPAN Stacks: A Survey." *7th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2011*: 1–4.
- Dey, Nilanjan, Amira S. Ashour, and Chintan Bhatt. 2017. "Internet of Things Driven Connected Healthcare." In Springer, Cham, 3–12.
http://link.springer.com/10.1007/978-3-319-49736-5_1 (April 30, 2019).
- Dunkels, Adam et al. 2011. "The Contiki Os: The Operating System for the Internet of Things." *Online*, at <http://www.contikios.org> 605.
- Gaddour, Olfa, and Anis Koubâa. 2012. "RPL in a Nutshell: A Survey." *Computer Networks* 56(14): 3163–78.
- Ge, Shu Yuan, Seung Man Chun, Hyun Su Kim, and Jong Tae Park. 2016. "Design and Implementation of Interoperable IoT Healthcare System Based on International Standards." *2016 13th IEEE Annual Consumer Communications and Networking Conference, CCNC 2016*: 119–24.
- Ghosh, Ananda Mohon, Debashish Halder, and S. K. Alamgir Hossain. 2016. "Remote Health Monitoring System through IoT." *2016 5th International Conference on Informatics, Electronics and Vision, ICIEV 2016*: 921–26.
- Hassanalieragh, Moeen et al. 2015. "Health Monitoring and Management Using Internet-of-Things (IoT) Sensing with Cloud-Based Processing: Opportunities and Challenges." *Proceedings - 2015 IEEE International Conference on Services Computing, SCC 2015*: 285–92.
- He, Daojing et al. 2018. "Privacy in the Internet of Things for Smart Healthcare." *IEEE Communications Magazine* 56(4): 38–44.
- Imane, Sahmi, Mazri Tomader, and Hmina Nabil. 2019. "Comparison between CoAP and MQTT in Smart Healthcare and Some Threats." *International Symposium on Advanced Electrical and Communication Technologies, ISAECT 2018 - Proceedings*: 1–4.
- Instruments, Texas. 2015. "Cc2538 Powerful Wireless Microcontroller System-on-chip for 2.4-ghz IEEE 802.15. 4, 6lowpan, and Zigbee Applications." *CC2538 datasheet (April*

- 2015).
- Irman, Muhammad. 2018. "Low Cost Heart Rate Portable Device for Risk Patients with LoT and Warning System." : 46–49.
- Islam, S. M.Riazul et al. 2015. "The Internet of Things for Health Care: A Comprehensive Survey." *IEEE Access* 3: 678–708.
- El Kafhali, Said, and Khaled Salah. 2018. "Performance Modelling and Analysis of Internet of Things Enabled Healthcare Monitoring Systems." *IET Networks* 8(1): 48–58.
- Khattak, Hasan Ali, Michele Ruta, and Eugenio Di Sciascio. 2014. "CoAP-Based Healthcare Sensors Network: A Survey." *Sisinflab.Poliba.It*.
http://sisinflab.poliba.it/publications/2014/KRD14/khattak_et_al_ibcast2014.pdf.
- Kortuem, Gerd, Fahim Kawsar, Daniel Fitton, and Vasughi Sundramoorthy. 2010. "Smart Objects as Building Blocks for the Internet of Things." *Internet Computing, IEEE* 14(1): 44–51.
- Koutsouris, Nikos, Katerina Giannakopoulou, and Vanessa De Luca. 2018. "InLife : A Platform Enabling the Exploitation of IoT and Gamification in Healthcare." *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*: 224–30.
- Kovatsch, Matthias, Martin Lanter, and Zach Shelby. 2014. "Californium: Scalable Cloud Services for the Internet of Things with Coap." In *2014 International Conference on the Internet of Things (IOT)*, IEEE, 1–6.
- Kumar, Priyan Malarvizhi, and Usha Devi Gandhi. 2017. "Enhanced DTLS with CoAP-Based Authentication Scheme for the Internet of Things in Healthcare Application." *Journal of Supercomputing*: 1–21.
- Li, Shancang, Li Da Xu, and Shanshan Zhao. 2015. "The Internet of Things: A Survey." *Information Systems Frontiers* 17(2): 243–59.
- Lo, Benny P.L., Henry Ip, and Guang Zhong Yang. 2016. "Transforming Health Care: Body Sensor Networks, Wearables, and the Internet of Things." *IEEE Pulse* 7(1): 4–8.
- Mainetti, Luca, Luigi Patrono, and Antonio Vilei. 2011. "Evolution of Wireless Sensor Networks towards the Internet of Things: A Survey." In *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference On*, , 1–6.
- Martínez-Caro, Eva, Juan Gabriel Cegarra-Navarro, Alexeis García-Pérez, and Monica Fait. 2018. "Healthcare Service Evolution towards the Internet of Things: An End-User Perspective." *Technological Forecasting and Social Change* 136(March): 268–76. <https://doi.org/10.1016/j.techfore.2018.03.025>.
- Muhammad, Ghulam, SK Md Mizanur Rahman, Abdulhameed Alelaiwi, and Atif Alamri. 2017. "Smart Health Solution Integrating IoT and Cloud: A Case Study of Voice Pathology Monitoring." *IEEE Communications Magazine* 55(1): 69–73.
- Nausheen, Farha, and Sayyada Hajera Begum. 2018. "Healthcare IoT : Benefits , Vulnerabilities and Solutions." *2018 2nd International Conference on Inventive Systems and Control (ICISC) (Icisc)*: 517–22.
- Österlind, Fredrik. 2006. "A Sensor Network Simulator for the Contiki OS." *SICS Research Report*.
- Qi, Jun et al. 2017. "Advanced Internet of Things for Personalised Healthcare Systems: A

- Survey.” *Pervasive and Mobile Computing* 41: 132–49.
<http://dx.doi.org/10.1016/j.pmcj.2017.06.018>.
- Qi, Jun, Po Yang, Dina Fan, and Zhikun Deng. 2015. “A Survey of Physical Activity Monitoring and Assessment Using Internet of Things Technology.” *Proceedings - 15th IEEE International Conference on Computer and Information Technology, CIT 2015, 14th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2015, 13th IEEE International Conference on Dependable, Autonomic and Se*: 2353–58.
- Sehgal, Anuj. 2013. “Using the Contiki Cooja Simulator.” *Computer Science, Jacobs University Bremen Campus Ring 1*: 28759.
- Shelby, Z., K. Hartke, and C. Bormann. 2014. *The Constrained Application Protocol (CoAP)*.
- Swaroop, K. Narendra, Kavitha Chandu, Ramesh Gorrepotu, and Subimal Deb. 2019. “A Health Monitoring System for Vital Signs Using IoT.” *Internet of Things* 5: 116–29.
<https://doi.org/10.1016/j.iot.2019.01.004>.
- Tran, Van Loc, Anik Islam, Jeevan Kharel, and Soo Young Shin. 2018. “On the Application of Social Internet of Things with Fog Computing: A New Paradigm for Traffic Information Sharing System.” *Proceedings - 2018 IEEE 6th International Conference on Future Internet of Things and Cloud, FiCloud 2018*: 349–54.
- Ugrenovic, Dejana, and Gordana Gardasevic. 2016. “CoAP Protocol for Web-Based Monitoring in IoT Healthcare Applications.” In *2015 23rd Telecommunications Forum, TELFOR 2015*, , 79–82.
- Vilajosana, Xavier, Pere Tuset, Thomas Watteyne, and Kris Pister. 2015. “OpenMote: Open-Source Prototyping Platform for the Industrial IoT.” In *International Conference on Ad Hoc Networks*, Springer, 211–22.
- Wan, Jie et al. 2018. “Wearable IoT Enabled Real-Time Health Monitoring System.” *Eurasip Journal on Wireless Communications and Networking* 2018(1).
- Watteyne, Thomas et al. 2012. “OpenWSN: A Standards-based Low-power Wireless Development Environment.” *Transactions on Emerging Telecommunications Technologies* 23(5): 480–93.
- Wayangankar, Sravani Ganesh, and Priti Prakash Jorvekar. 2018. “Survey on Internet of Things in the Fog.” *2018 IEEE Global Conference on Wireless Computing and Networking (GCWCN)*: 80–86. <https://ieeexplore.ieee.org/document/8668610/>.
- Yang, Yang, Ximeng Liu, and Robert H. Deng. 2018. “Lightweight Break-Glass Access Control System for Healthcare Internet-of-Things.” *IEEE Transactions on Industrial Informatics* 14(8): 3610–17.
- YIN, Yuehong, Yan Zeng, Xing Chen, and Yuanjie Fan. 2016. “The Internet of Things in Healthcare: An Overview.” *Journal of Industrial Information Integration* 1: 3–13.
<http://dx.doi.org/10.1016/j.jii.2016.03.004>.