JIMMA UNIVERSITY

JIMMA INSTITUTE OF TECHNOLOGY

FACULTY OF COMPUTING

Remote and Flexible User Application Deployment Using Web Services for Constrained Networks

By: Zerihun Befekadu

Advisor: Girum Ketema (PhD)

Co-Advisor: Worku Birhanie (MSc)

A THESIS SUBMITTED TO:

THE FACULTY OF COMPUTING OF JIMMA UNIVERSITYIN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER NETWORKING

Jimma, Ethiopia

June, 2019 G.C

JIMMA UNIVERSITY

JIMMA INSTITUTE OF TECHNOLOGY

FACULTY OF COMPUTING

Remote and Flexible User Application Deployment Using Web Services for Constrained Networks

By: Zerihun Befekadu

Advisor: Dr. Girum Ketema (PhD)

Co-Advisor: Worku Birhanie (MSc)

Approved by:

Board of Examiners:	**Signature**
1 Asrat Mulatu (PhD)	_____
2. Mr. Gemechu Birhanu (Msc)	_____

Acknowledgment

First of all I would like to thank God for His support to accomplish this thesis and being with me in all directions of my life.

I would like to give special thanks to my advisor, Dr. Girum Ketema, for the time and energy He has invested on me throughout the work. This work would not have been possible without his support, guidance, invaluable feedback and advice.

I would like to thank also my co-advisor Worku Birhanie for commenting the thesis proposal and confirming it as possible.

Finally, I take this opportunity to express my gratitude to my family and to all my friends who have supported me during the various aspect of conducting this thesis.

Thank you!!

Dedication

To: Barkot.

Table of Content

List of Tables

List of Figures

List of Acronyms

IoT	Internet of Things
WSN	Wireless Sensor Network
CoAP	Constrained Application Protocol
URI	Uniform Resource Identifier
DTLS	Datagram Transport Layer Security
UDP	User Datagram Protocol
RF	Radio Frequencies
OS	Operating System
NesC	Network embedded systems C
EEPROM	Electrically Erasable Programmable Read Only Memory
RAM	Random Access Memory
IETF	Internet Engineering Task Force
MTU	maximum transmission unit
CSMA-CA	Multiple Access with Collision Avoidance
RPL	Routing Protocol for Low Power and Lossy Networks

Abstract

Constrained networks or Low Power and Lossy Network (LLN) are networks that consists of spatially distributed autonomous sensors, which are tiny devices deployed to cooperatively process, communicate and monitor the physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and pass their data through the network to a main location.

Constrained sensor nodes, in addition to their resource constrained (energy, processing power, memory, and communication bandwidth), are often placed in a hard-to-reach location in harsh and extreme environments, because of that, it is very difficult to maintenance and repair. Sensor nodes must be self-managing meaning they must be able to autonomously configure, update them, cooperate with other nodes and accommodate to failures and environmental changes without human intervention. Considering such sensor deployment, LLNs are envisioned to be deployed in the absence of permanent network infrastructure and in environments with limited or no human accessibility.

In this thesis, we design a mechanism to aggregate data generated by multiple sensors in the constrained networks using Constrained Application Protocol (CoAP) and send it to the next hop node which may do further aggregation. In addition, it devises a method that can be remotely deploying aggregation module to a remote node using Constrained Application Protocol block-wise transfer. It is implemented in Contiki with COOJA simulator and evaluated the implementation by taking different measures such as energy consumption, delay and number of packet transmitted in the network.

This solution makes flexibility to access and exchange sensor nodes application components and minimize the number of packets in network which reduce battery consumption of each sensor nodes and communication energy of overall constrained network.

Key words: *LLN, CoAP, Cooja, Contiki, Constrained Network.*

Chapter One: Introduction

1.1 Background

Internet of Things (IoT), an emerging topic of technical, social, and economic significance, that connect our world (things) more than we ever thought possible all our infrastructure systems forming a network called constrained network, or Low Power and Lossy Network (LLN) by using constrained sensor devices.

Low Power and Lossy Network consists of spatially distributed autonomous sensors, which are tiny devices deployed to cooperatively process, communicate and monitor the physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and pass their data through the network to a main location.

Constrained sensors are deployed in harsh and extreme environment, because of that, it is more vulnerable to different types of security issues, there is no possibility of maintenance and repair in remote areas, in harsh environments sensor nodes must be self-managing which means they must be able to autonomously configure, update themselves, cooperate with other nodes and accommodate to failures and environmental changes without human intervention. So that Low Power and Lossy Network (LLN) are envisioned to be deployed in the absence of permanent network infrastructure and in environments with limited or no human accessibility

Specifying the application component or program ranging from full image replacement to virtual machines after deployment and changing it during operation has become necessary, since the precise application requirements and processing methods are often not fully known until a sensor network is actually deployed. Thanks to researchers and the evolution of technologies there are a variety of mechanisms that exist today to deploy new application or to fix bugs in deployed systems, of course such deployments demand mechanisms to update nodes over the wireless network: collecting nodes to deploy updates, e.g., from simple modifications to the introduction of new functionality or re-tasking of sensor nodes, is often tedious [1], or even dangerous [2]. Remote image replacement or application deployment schemes for sensor nodes meet challenges like resource consumption i.e. energy, processing power, memory, and communication bandwidth are scarce resources on sensor nodes and their consumption needs to be limited, and

integration those new application component or full image into the system architecture also another challenge.

Flexible remote deployment or replacement of any image or application component always considered as the limitation of the constrained network resource and integration of the new component to the system architecture.

The focus of this paper is to design and develop a new user application module that enables nodes to do data pre-processing locally and deploy the application module to a constrained sensor node remotely over a wireless link (from anywhere), this makes the networked nodes accessibility more flexible and easy to reprogram in environments especially where sensor nodes are deployed in limited human accessibility areas. The proposed approach decrease the overall constrained network data transmission by twofold

1) Enable in-network-processing and hence reduce number of unnecessary packet transmissions on the way to reach to the Base Station.
2) It makes the network management easy by doing a flexible remote deployment via a constrained network without disturbing the network.

We use a CoAP URI Query to register the sensor node to aggregator sensor node and the aggregator node initiates the new module application to compute the average value and send it to the next hop node which may do further aggregation processing. CoAP block transfer based, flexible protocol is used to deploy the new user application module. Here, the Base Station or any other smart device like laptop can initiate the process of remote deployment of the application module whenever the new functionalities module need to be added, wants to upgrade the application version or wants to change the application itself. The simulation used to work for this thesis is Contiki constrained network simulator.

1.2 Motivation and Statement of the problem

Low Power and Lossy Network, in addition to their resource constrained, are deployed in harsh and extreme environments, because of which, there is no possibility of maintenance and repair such sensor nodes must be self-managing meaning they must be able to autonomously configure, update themselves, cooperate with other nodes and accommodate to failures and environmental changes without human intervention. Considering such sensor deployments, constrained networks are envisioned to be deployed in the absence of permanent network infrastructure and in environments with limited or no human accessibility.

This thesis proposed a flexible and easier solution to make easily accessible to the components of a constrained sensor nodes, sensor nodes which are deployed in harsh environment, using a remote user application deployment to the constrained sensor node via a constrained network.

The new user application module, which is remotely deployed to sensor node from the Base Station or laptop or mobile, will aggregate the generated data from sensor nodes and can send simple pre-processing tasks such as averaging, finding minimum or maximum values using In-Network-Process then it sends the average value to the Base Station so that only the minimum required data value will be transmitted over the constrained network communication media.

1.3 Objectives

1.3.1 General Objective

The general objective of this work is to provide remote and flexible user application deployment using web services for constrained networks node without disturbing the other sensor nodes on the network and limiting unnecessary/unwanted data transmission over the constrained network media.

1.3.2 Specific Objectives

The specific objectives of this work are listed as follows:

- ✓ Design and develop a new user application module,
- ✓ Flexible remote user application deployment mechanism based on CoAP Block transfer protocol from Base Station to constrained sensor node over the constrained network,
- ✓ Sensor nodes subscription to aggregator sensor nodes using web based URI-Query
- ✓ Perform functional and performance evaluations to see effectiveness of the new approach using different metrics like number of packets to complete the entire communication and communication energy consumption with that of the existing system.

1.4 Methodology

1.4.1 Methods

To accomplish the objectives of this proposed thesis work, the following steps are followed:

1.4.2 Literature review

In order to achieve the objectives of this thesis various related resources like books, published research papers and other documents are revised to get accurate knowledge, implementation about wireless sensor network, user application deployment.

1.4.3 Software and Simulation tools

The following Software and simulation tools, that are required for development and simulation, are identified and studied:

- ✓ **Contiki OS** is a lightweight open source Operating System written in C for constrained networks. Contiki is a highly portable OS and it is built around an event-driven kernel.

Contiki provides preemptive multitasking that can be used at the individual process level [3]. Also it is an open source operating system for the Internet of Things (IoT). Contiki connects tiny low-cost, low-power microcontrollers to the Internet. Contiki is a powerful toolbox for building complex wireless systems.

- ✓ **CoAP** Constrained Application Protocol is a Restful transfer protocol for constrained nodes and networks. Basic CoAP messages work well for the small payloads we expect from temperature sensors, light switches, and similar building-automation devices. [4]. Cooja is the Contiki network simulator allows large and small networks of Contiki motes to be simulated. Motes can be emulated at the hardware level or which is faster and allows simulation of larger networks.[5] We choose CoAP in our thesis work because it is designed to easily interface with HTTP for integration with the Web while meeting specialized requirements such as multicast support, very low overhead, and simplicity for constrained environments.
- ✓ **CoAP Block-wise transfer** an extension used in CoAP that allow limiting the size of datagram in constrained network. CoAP is based on datagram transports such as UDP or Datagram Transport Layer Security (DTLS), the maximum size of resource representations that can be transferred without too much fragmentation is limited. The Block options provide a minimal way to transfer larger resource representations in a block-wise fashion. The size of the datagram used in Block size transfer in constrained networks is maximum datagram size (~ 64 KiB for UDP). Block options to CoAP that can be used for block-wise transfers. Benefits of using these options include:
    - Transfers larger than what can be accommodated in constrained network link-layer packets can be performed in smaller blocks.
    - The transfer of each block is acknowledged, enabling individual retransmission if required.
    - No hard-to-manage conversation state is created at the adaptation layer or IP layer for fragmentation.
- ✓ **Performance Evaluation:** After the system is fully implemented, in order to validate energy improvement and effectiveness of the designed constrained network as well as its proper functioning and overall energy consumption improvement is checked using the following metrics:
    - Number of packet transmitted in constrained networks in the way of

communication to gateway node

- Flexibility of dynamic deployment
- Communication energy improvement

1.5 Scope

This thesis work focuses to design, a user application module and remote deployment of a new user application module remotely to sensor node over a constrained network and this application module improve the communication energy, by decreasing the number of packet transmission thought out the entire constrained network compared with that of the existing system paradigm, using simulator tool due to unavailability of real sensor node and some other required device.

Our proposed work will not cover the following tasks:

- ✓ No real sensor node and sensor network is used due to unavailability of the motes in the market
- ✓ Automatic sensor node selection for deploying the new designed module is not applied.
- ✓ Data compression techniques to overcome the bandwidth limitation of constrained networks.
- ✓ No authenticate or hashing or integrity is applied while transfer the new user application.

1.6 Organization of the study

This thesis work comprises of six chapters. The next Chapter covers the study of a constrained network, the standards, the architecture, hardware that are related to constrained network and applications. Chapter 3 discusses related works that have significant relation with this thesis. Even if there are a number of works done on this area, we select the most related works to our thesis. Chapter 4 shows the designed and the proposed work. Chapter 5 discusses the development of the proposed system implementation and simulation using a powerful constrained network emulation and simulation software tools called contiki. Finally, chapter 6, we summarize the contributions made in the thesis, and conclude our work based on the results obtained from the thesis. Furthermore, new issues that have been surfacing while working on the thesis will be suggested as future work.

Chapter Two: Literature Review

2.1 Constrained networks

Internet of things are a new class of distributed systems [6] where a collection of nodes are organized in a cooperative network intended to monitor the physical condition and to cooperatively pass the data through an ad-hoc network to the main location. One of the main components in a constrained network is sensor nodes, each of which is usually made up of a microcontroller, antenna, transceiver, memory, power source and one or more sensors. A typical wireless sensor network is shown in Figure 2.1

Zolertia Z1 Mote (which is used in our thesis and described in detail in section 5.1.1) is a general purpose development platform for constrained network designed for researchers, developers, enthusiasts and hobbyists [7]. Its transceiver is based on the IEEE 802.15.4 protocol [8].

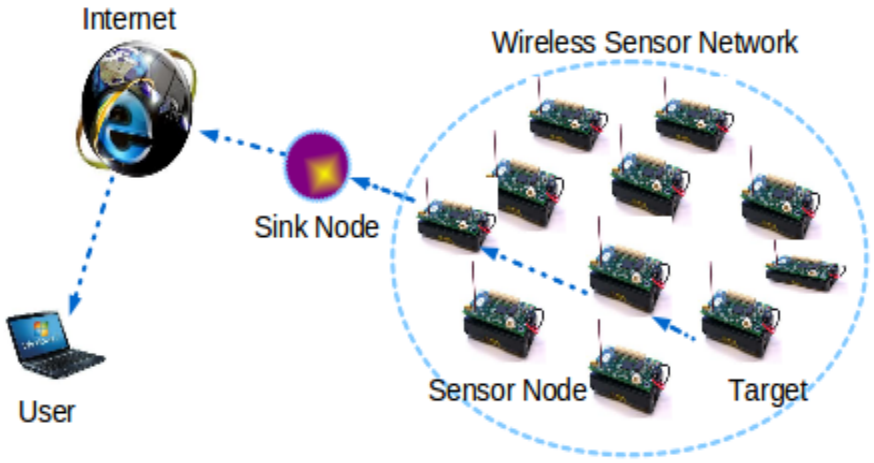

Figure 2.1: Constrained network

2.2 Characteristics of constrained sensor networks

A brief description of a constrained network characteristics are described as follows:

Dynamic Network Topology: It is important to Re-deployment the network topology in case of failure of the node, failure of radio links, or arrival of some mobile obstacles.

Limited power: In some environment it is not easy to charge energy regularly [9]. They may, probably, be. So, energy consumption is a major issue, and should be optimized at three stages, node communication, sensing and processing to reduce the energy consumption.

Node mobility: It makes the network links to be formed dynamically, that more nodes can join to the network easily or disjoin when they move out of the range.

Unattended operation: Ability of reconfiguration the network by the nodes without any human intervention

Large scale of deployment: The large scale of constrained network from hundreds or thousands of nodes and some environmental parameters like noise, dispersion, interface and available bandwidth, effects on the connection quality and may causes some disconnection between the nodes even in tiny networks [10].Advantages and disadvantages of constrained networks are listed in the table 1 [11]

Table 2.1: Constrained Network advantages and disadvantages

Advantages	Disadvantages
Low cost implementation	More complex to configure than a wired network
Could be set up in the non-reachable places	Lower speed
No need the fixed infrastructure to set up the network	Less secure
Flexible if there is ad hoc situation when additional workstation is required	Easily affected by surroundings (walls, Microwave, large distances due to Signal attenuation).

2.3 Constrained Senor Nodes

A sensor node, also known as a mote, is a node in a constrained network that is capable of performing data processing, gathering sensory information, communicating with other connected nodes in the network over flexible network architecture [12]. It consists of computation, sensing and communication units and are often deployed in hostile environments or over large geographical areas, and have been developed and used in various different fields, from indoor to outdoor. Building a constrained network first of all requires the constituting nodes to be developed and made available.

These nodes have to meet the requirements that come from the specific requirements of a given application: they might have to be small, cheap, or energy efficient, they have to be equipped with the right sensors, the necessary computation and memory resources, and they need adequate communication facilities.

Sensor nodes are designed focusing on lower energy consumption and easier development process for a given wireless communication range and area. Hence they are categorized in two different classes [13].One is ordinary sensor nodes which are used to sense different physical phenomena, and the other is the sink/gateway node that connects sensor networks to the Internet. In this thesis, sensor nodes are used to communicate with each other within the constrained network and send data to the sink, so we use both of them because they are available on Cooja constrained sensor network simulator by using Contiki operating system which is described letter in chapter 5.

A typical constrained sensor node is shown in Figure 2.2

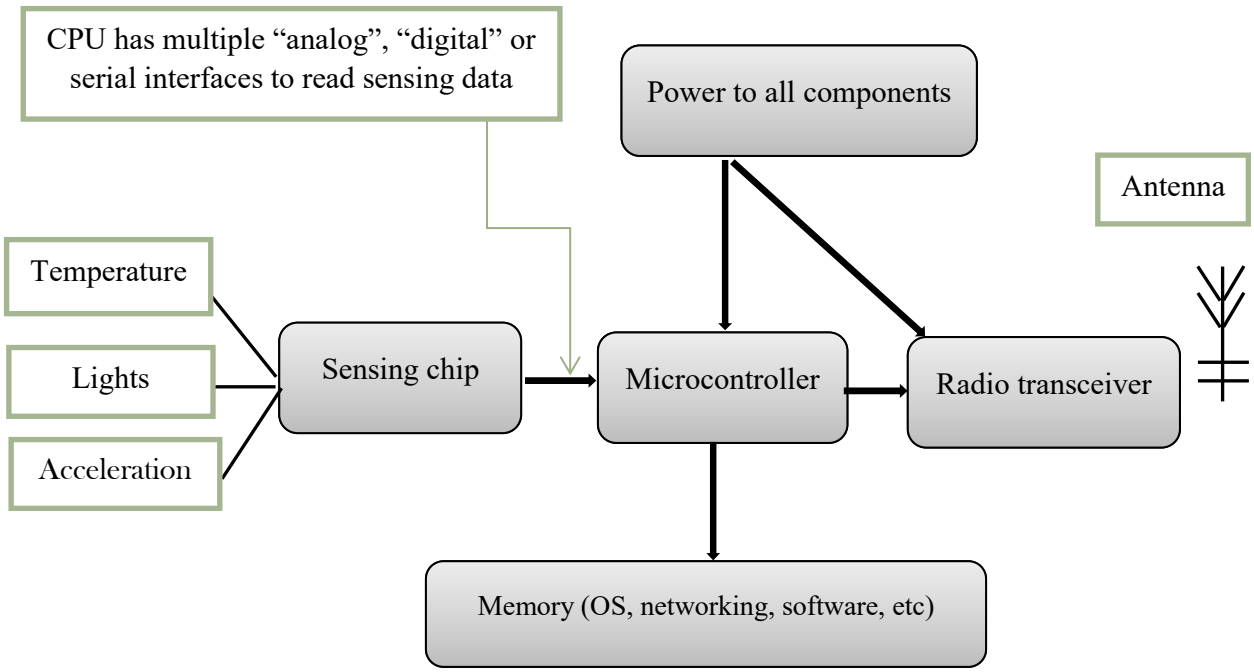

Figure 2.2: Components of constrained sensor node [14]

The above different components are described as follows [15].

Microcontroller/Processor: is the core of a sensor node which is in charge of processing data and executing the code that describes the operation of the sensor node. The processor gets data from the sensors, processes this data, decides when and where to send it, receives data from other sensor nodes, and decides on the actuator’s behavior. It has to execute various programs, ranging from time-critical signal processing and communication protocols to application programs. System-on-chip (SOC) technology enables integrating a complete system on a single chip. Commercial SOC based embedded processors from Atmel, Intel, and Texas Instruments have been used for sensor nodes such as UC Berkeley's nodes.

Memory: is used to store programs and data. Commonly random access memory (RAM) is used to store intermediate and temporary samples or packets from other nodes. Memories like Electrically Erasable Programmable Read Only Memory (EEPROM) are used to store the program code. However, the energy consumption by memory is an important factor for the appropriate design of the sensor node.

Radio Transmitter: is the hardware to enable the networking capability of the sensor nodes. Some usual methods for communication between the sensor nodes are Radio Frequencies (RF), optical communication, etc. RF is generally used because it provides a long range of transmission and reception at high rate with acceptable error rates for the required energy and also it does not require a direct line of sight between two neighbors.

Sensing chip: A sensor is an electronic component that measures the physical quantity and converts to the type that can be read by the user or other electronic device such as, a microprocessor. Micro Electro Mechanical System (MEMS) technology is now available to integrate a rich set of sensors onto the same chip. They are interfaces to the physical world which sense the environment parameter and convert them to a raw data (mostly voltage or current) for further processing within the processor. Commercially available sensors include thermal, acoustic/ultrasound, and seismic sensors, magnetic and electromagnetic sensors, optical transducer, chemical and biological transducer, accelerometer, and barometric pressure detectors. These sensors can be used in a broad range of applications. Commercially there are readily available sensors for different applications. [16]

Power Supply: For sensor nodes, the power supply is a crucial system component. There exists a large quantity of power supply options for a sensor node. The most common option is the use of batteries; other options are scavenging energy from the environment where the sensor node is exposed, the most popular example is solar cell. The integration of the above technologies has made it possible to integrate sensing, computing, communication, and power components in to tiny sensor nodes.

2.4 Constrained Network Communication Standard

Standardization bodies, IEEE, have developed standards for the Internet of Things and IP protocol stack for Low-Power, Reliable constrained sensor networks have defined. The different layers and their protocols are listed in table 2.2 and which will be described in next sections.

Table 2.2: IP protocol stack for Low Power, Reliable WSN

Layer	Protocols
Transport/Application	CoAP
Network/Routing	RPL, BCP, CTP
Adaption	6LoWPAN
MAC	IEEE 805.15.4.E, B-MAC
Physical	IEEE 802.15.4

2.4.1 Application Layer: CoAP

Constrained Application Protocol (CoAP) is an application layer protocol, a web transfer protocol for constrained nodes and constrained networks like IoT, WSN, and M2M, to be used in very simple electronics devices that allow them to communicate interactively over the Internet [19]. It is particularly targeted for small low power sensors, switches, valves and similar components that need to be controlled or supervised remotely, through standard Internet networks. CoAP attempts to tackle different problems to work in a suitable way for constrained nodes and networks some of the solutions are as follows:

- CoAP removes the overhead and complexity of TCP by operating entirely over UDP and implementing an optional acknowledgements system. This allows for messages to be transmitted with only best-effort and in cases of network congestion, a sensor reading will be lost instead of adding retransmission attempts into congested network.
- CoAP uses a smaller amount of data transition.

  TCP requires a three-way handshake before any communication can even begin however this is not necessary with CoAP as data can be sent on the first packet. In constrained network, the client may not require a reply and so does not need to store any state about the connections and can simply discard incoming packets. The amount of data to be

transmitted is further reduced by using a binary header as opposed to the ASCII header used by HTTP.

- CoAP use with multicast, which would allow sensor nodes to send their updates to a multicast group as oppose to a single server. This can be used for a server to simply listen to a multicast group.

Summary of CoAP features

Table 2.3: CoAP features

CoAP Features
✓ It is open IETF standard
✓ Asynchronous transaction model
✓ Embedded web transfer protocol (coap://)
✓ UDP binding with reliability and multicast support
✓ URI support
✓ GET, POST, PUT and DELETE methods are used
✓ Small, sample 4 byte header
✓ Supports binding to UDP, SMS and TCP
✓ DTLS based PSK,RPK and Certification security
✓ Built-in discovery
✓ Optional observation and block transfer

From the above analyses we can summarize the usefulness (listed below) and its best features (table 2.3 CoAP features) of CoAP transfer protocol, is best to use for a constrained networks and the reason to choose for implementation in our thesis.

- ✓ It is clear that, CoAP is becoming more and more suitable for use in the constrained environments that are found in sensor nodes.
- ✓ CoAP stack is also clearly a better solution than HTTP stacks for constrained networks but it seems that there are still places where it has shortcomings.
- ✓ CoAP would be useful for tasks such as pushing configuration updates to nodes and receiving confirmations of completion using a multicast request.

❖ **CoAP vs. HTTP**

The model of CoAP is similar to the client server model of HTTP. CoAP web transfer uses similar methods as HTTP when sending requests from clients to servers, namely PUT, POST, GET, and DELETE. In client server communication, the CoAP client sends a request to a specific resource on a server by using one the methods and the server responds with the respective response. Figure 2.3 shows a typical request response interaction between a CoAP client and server. CoAP is based on REST architecture, which is a general design for accessing Internet or machine to machine resources thus, result in a CoAP implementation acting in both client and server roles.

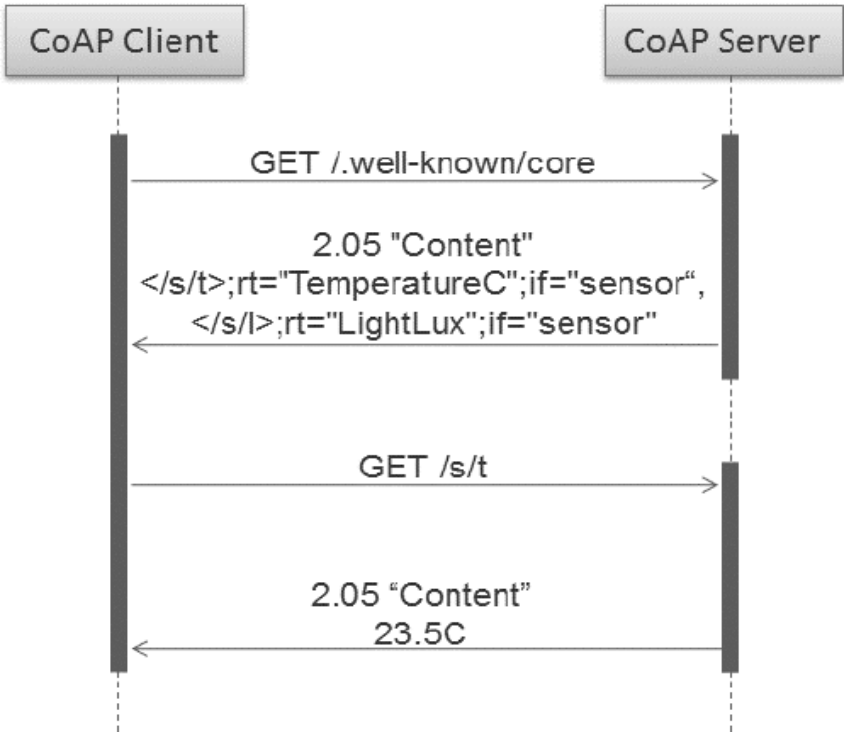

Figure 2. 3: CoAP Operation [42]

Unlike HTTP based protocols, CoAP operates over UDP instead of using complex congestion control as in TCP [20]. In order to overcome disadvantage in constrained resource, CoAP need to optimize the length of datagram and provide reliable communication. CoAP intends to avoid any complexity by running over UDP instead of TCP that has complexity in congestion control. Figure 2.4 shows the HTTP and CoAP protocol stacks.

Figure 2. 4: HTTP and CoAP protocol stacks

- **CoAP Model**

CoAP application layer transfer protocol model, Figure 2.5, works in a two layers structure namely,

1. **Message layer***:* is the first layer designed to communicate with UDP and asynchronous switching. It supports 4 types message: ***CON (confirmable)*** *a request or response and require an Acknowledgment.* ***NON (non-confirmable)*** *a regularly repeated messages that doesn't require an Acknowledgment.* ***ACK (Acknowledgement)*** *type of message that carry a response or sometimes it could be empty [Code field value is 0 in CoAP header].* ***RST (Reset)*** *a message sent in case a CON message is not received properly.*
2. **Request/response layer:** is the second layer concerns communication method and deal with request/response message. ***Piggy-backed:*** *Client sends request using CON type or NON type message and receives response ACK with confirmable message immediately.* ***Separate response:*** *If server receive a CON type message but not able to response this request immediately, it will send an empty ACK in case of client resend this message.* ***Non confirmable request and response:*** *client send NON type message indicate that Server don't need to confirm. Server will resend a NON type message with response*

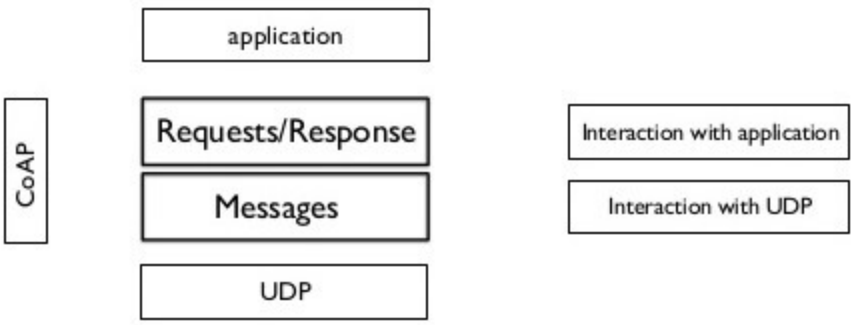

Figure 2. 5: CoAP layering Model

2.4.2 CoAP: Message Format

CoAP uses simple binary format which is based on the exchange of compact messages that are transmitted over UDP. You can see a typical CoAP message format at Figure 2.6.

2b	**2b**	**4b**	**1B**	**2B**
Ver	TKL	OC	Code	Message ID
Token (if any)…				
Options (if any)…				
11111111			Payload (if any)…	

Figure 2. 5: CoAP message format

- Where:
  - Ver – CoAP version(2 bit)
  - TKL – Indicates length of token(2 bit)
  - OC – Option count(4 bit)
  - Code – Request method (1-10) or Response method (40-255)(8 bit)
    - GET: 1
    - POST: 2
    - PUT: 3
    - DELETE: 4
  - Message ID – Unique Identifier for matching response(16 bit)

2.4.3 Routing: Low power and Lossy Networks (RPL)

Routing is a fundamental operation in network communication, which deals with packet delivery from source to destination. [21] Low-power and lossy networks (LLN) is composed of many embedded devices, typically, the energy of these embedded devices itself, processing, and storage capabilities are limited, the embedded device via Bluetooth, IEEE 802.15.4, or Wi-Fi and other wireless technology to connect to the instrument [22] Low-power networks are often easy to lose no predefined topology. For such low-power and lossy network, an IPv6 Routing Protocol

for Low power and Lossy Networks (RPL) is very suitable which is developed by IETF ROLL working group in order to overcome such routing issue. RPL routing protocols mitigates this issue by connecting automatically discover and create and maintain the topology. [23] RPL routing protocol node through the exchange of DIS, DIO, ICMPv6 and DAO control messages, thus creating the topology and routing, RPL routing protocol establish divided into the following two processes: topology to establish and build up the route, the route down the establishment. Down route established when there are two modes: ***Non-Storing Mode*** and **Storing Mode.** [24]

- ✓ ***Destination Oriented Directed Acyclic Graph (DODAG):*** which is routed at a single destination and constructed by RPL in order to avoid any cycle in the connected nodes. The logical topology of the network will be defined by his graph.
- ✓ ***DODAG Information Object (DIO)*** message is sent periodically by the root node down to the leaf nodes to form and maintain the DODAG graph
- ✓ ***Destination Advertisement Object (DAO)*** is sent from nodes to their parents to inform their presence.
- ✓ ***DODAG Information Solicitation (DIS)*** message is used by nodes to enforce other nodes in the network to send DIO messages.[25]

2.4.4 Adaptation: 6LoWPAN

Future IoT consisting of thousands of nodes and these networks may be connected to others via the internet. 6LoWPAN, a transport technique where LoWPAN is carried over IPv6, is defined by Internet Engineering Task Force (IETF) [26] to apply TCP/IP into constrained network [13]. In short the main function of the adaptation layer is to provide IPv6 and UDP header compression and fragmentation to transport IPv6 packets with a maximum transmission unit (MTU) [27]. Therefore, 6LoWPAN [28] [13] provides a means of carrying packet data in the form of IPv6 over IEEE 802.15.4 networks.

2.4.5 Media Access Control Layer: IEEE 802.15

The MAC Layer uses Multiple Access with Collision Avoidance (CSMA-CA) to detect whether or not another radio is transmitting and employ a method to avoid collisions. In this algorithm the MAC layer listens for energy or modulated data on the air. If none is detected, it can transmit immediately. If the channel is not clear the algorithm applies random wait times (back offs) before retrying the transmissions. Power management is another functionality of MAC layer that

allows radio devices to be turn off most of the time in order to save more energy. Contiki MAC [29] mechanism is used in contiki OS to replace the CSMA-CA.

2.4.6 Physical Layer: IEEE 802.15.4

The physical layer, which is defined by the IEEE 802.15.4, is the first layer in OSI reference model. The main different features of constrained network make the transmission at shorter distances, lower data rates and node power constraints. The physical layer is an important network stack layer to reduce energy dissipation by finding the optimal transmit (relay) distance and transmit power for a given modulation scheme and a given channel model, in order to maximize network lifetime. IEEE standard 802.15.4 aims to operate within a short range (i.e. 10 meters), with very low transmission rate of 250Kbit/s and with a reasonable battery life rather than other approaches, such as WI-FI, which offers more bandwidth and requires more power. The IEEE 802.15.4 packet consist of the 64-bit IEEE address or a short 16-bit address that is located in the destination and source addressing mode field, so the size of the packet could be different regarding to the address size. Figure 2.7 shows the general packet frame structure of IEEE 802.15.4.

The frame control field defines the type of the frame (i.e. data, acknowledgement or other type) and the addressing used (16 bit or 64 bit). The sequence number is an increasing counter of the frames transmitted by the node. The addressing fields contain the source and destination addresses. The following fields are the frame payload followed by the frame check sequence which ends the frame. The two subsections below are dedicated to the main functions of physical (PHY) layer and media access control (MAC) layer.

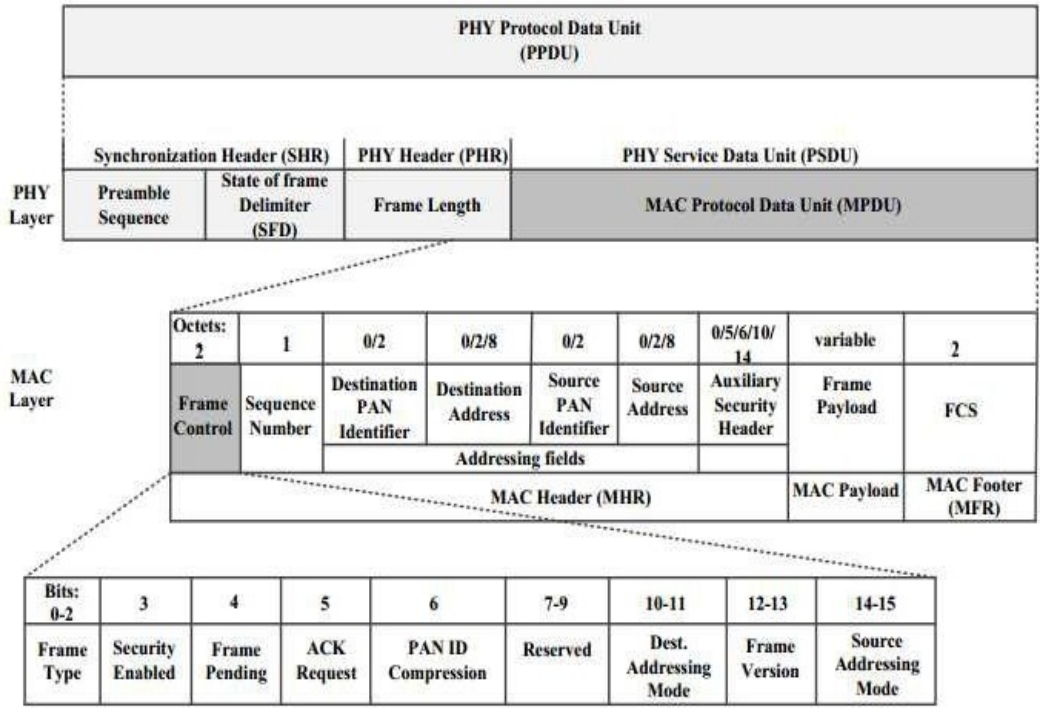

Figure 2.6: IEEE 802.15.4 General packet frame format

2.5 CoAP Block-wise Transfer

CoAP is based on datagram transports such as UDP or Datagram Transport Layer Security (DTLS), the maximum size of resource representations that can be transferred without too much fragmentation is limited, meaning the payload is considered to be as small as possible, this results very difficult to transfer large amount of data using CoAP transfer protocol. The CoAP Block-wise options provide a minimal way to transfer larger resource representations in a block-wise fashion instead of relying on IP fragmentation to overcome this kind of issues. In other word the block-wise transfer mechanisms are very fundamental to the use of CoAP for representations larger than about a kilobyte. When a resource representation is larger than the maximum size transferred in the payload of a single CoAP datagram, a Block option can be used to indicate a block-wise transfer. When a CoAP message is sent both with requests and with response the Block-wise provide two separate options to refer the transfer of the resource representation for each direction payload transfer, ***("Block1", "Size1")*** to the request i.e., a PUT or POST and ***("Block2", "Size2")*** for the response i.e., GET. This option indicates a block-wise

transfer and describes how this specific block-wise payload forms part of the entire body being transferred.

2.5.1 Structure of a Block Option

In transferring a Block (Block1 or Block2) option, the following three block options information's should be addressed,

- ✓ the size of the block (SZX),
- ✓ whether more blocks are following (M)
- ✓ the relative number of the block (NUM) within a sequence of blocks with the given size.

The value of the Block option is a variable-size (0 to 3 byte) unsigned integer. This integer value encodes these three fields, see Figure 2.8.

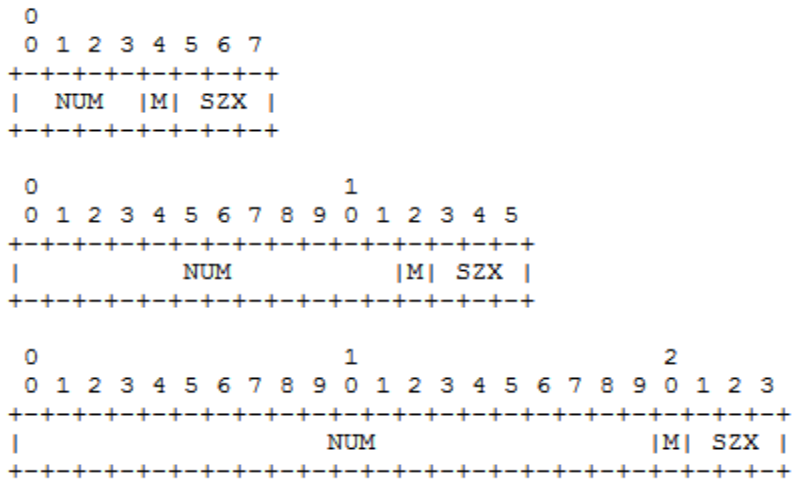

Figure 2. 7: Block Option Value

When all of NUM, M, and SZX happen to be zero, a zero-byte integer will be sent. The first field (NUM) indicates the block sequence number that "0" means it is the first block of this message. The M flag indicates the number of blocks are followed, M=0 means this is the last block of the message. The maximum size of this block defines in SZX field that is calculated using the formula Size = 2 ^ (Exp + 4). For example the SXZ=2 corresponds to a block size value of 64 bytes that the payload would be divided to 64 bytes and distributed to some blocks needed to transfer whole data. Note that the Block options support only a power of two block sizes from 16 to 1024 bytes.

Let us see two illustrative short examples, client server communication, with message flows for a block-wise GET, and for a PUT or POST that demonstrate the basic operation, the operation in the presence of retransmissions, and the operation of the block size negotiation.[30]

In Figure 2.9 client-server communication (***Block2 Option***), a block-wise GET with Early Negotiation illustration, the client sends a block size request based on the block-wise transfer link-format description. All ACK messages except for the last carry 64 bytes of payload.

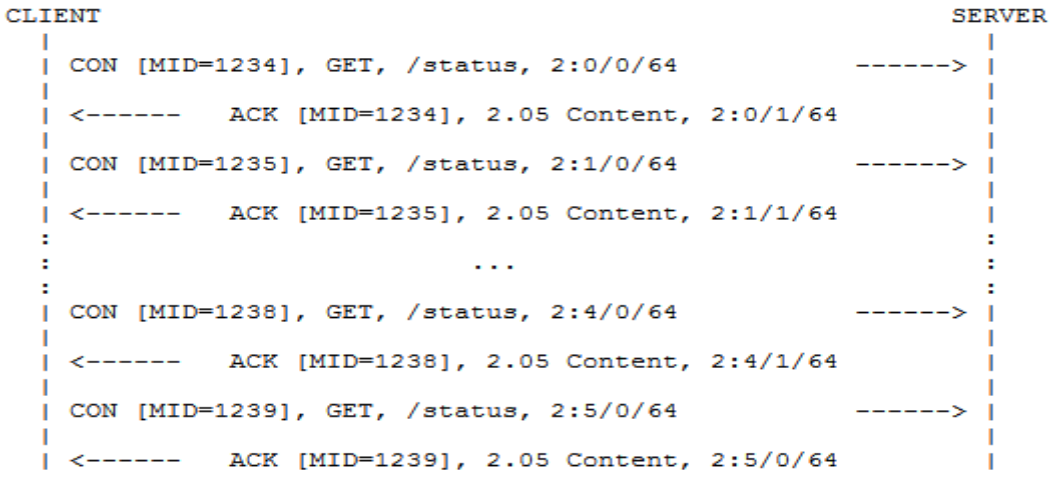

Figure 2. 8: Block-Wise GET with Early Negotiation

In Figure 2.10 client-server communication (***Block1 Option***), a block-wise PUT with Early Negotiation illustration, similar to GET, the responses to the requests that have a more bit in the request Block1 Option are provisional and carry the response code 2.31 (Continue); only the final response tells the client that the PUT succeeded.

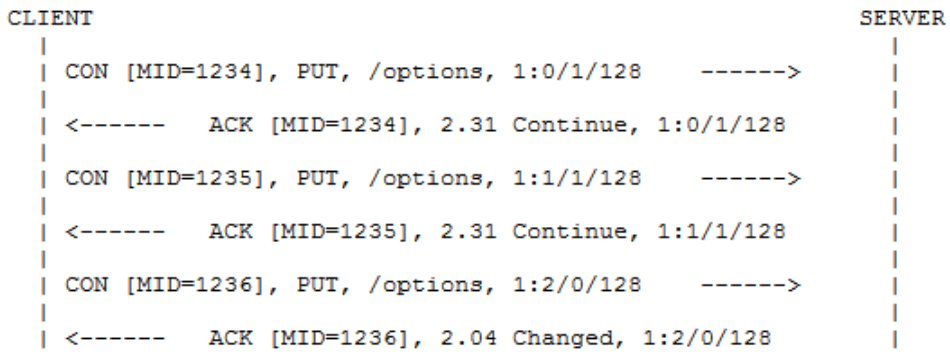

Figure 2. 9: Block-Wise PUT with Early Negotiation

Chapter Three: Review of Related Works

This chapter reviews some of the related works that have been published by different researchers. Most of the works are related to remote deployment of an application module (see section 3.1.) to a constrained sensor node remotely over a constrained network media and 2) section 3.2, are works that are related to In-network processing.

3.1 Remote deployment

Specifying the program after deployment and changing it during operation has become necessary, since the precise application requirements and processing methods are often not fully known until a sensor network is actually deployed. A range of possible solutions have been proposed allowing sensor nodes to be reprogrammed or deploy remotely in the field by using different approaches.

Range of approaches like in [34] - [38] based on a reprogramming of the constrained sensor node of the image by using the incremental code update mechanism which transmits an edit map encoding the differences between old and new program images. They generate the differences between the two files using a divide-and-conquer dynamic programming approach by adapting different algorithms that suit to their work. The incremental code update solution does not use block level code comparison, and hence is able to locate and send differences at byte-level granularity, by doing so they use the network traffic to send all the code updates continuously, which increase constrained network energy consumption by sending to many packets in constrained network.

M.B.Nirmala and A.S. Manjunath in [39] developed an efficient protocol to securely disseminate the code updates in constrained networks using mobile agents. Mobile agents are used to disseminate the code. Mobile agents traverse through the entire monitoring area of the network and disseminate the code. Mobile agents and also the code update are more vulnerable for adversaries, hence another measures are taken to detect the attacks and rectify them which is additional processing on the network. For example the sensor node should authenticate the mobile agents and check the integrity of the code. The code should be secured from adversaries, hence authentication, integrity, confidentially where the sensor node is certain about the origin of the program image and DoS are the security services need to be considered for secure code updates.

Munawar et al. present Dynamic TinyOS [33], which uses high-level knowledge of application structure to make application deployment. This is achieved using extensions to the NesC compiler, an extension to the C programming language designed to embody the structuring concepts and execution model, which convert TinyOS applications and system components into separate binary objects during compilation. Standard data dissemination protocols are then used to update individual objects. This approach also requires knowledge of program code structure, which reduces its applicability to systems developed using other compilers and languages.

Sensor Scheme as a novel interpreted WSN platform [40] for dynamically deployment sensor network applications. It is based on the semantics of the Scheme language and is equipped with high-level programming to improve the program's compactness. Hence the major design consideration of this work to improve the compactness of the program and memory allocation.

3.2 In-Network-Processing

T-Res programming [41] - written in Python, presented as a solution which models processing tasks as resources and make available on each constrained device and can be used by CoAP methods. Each T-Res resource stores URIs of the input and output devices as sub-resources. The last output and the compiled processing function are also stored as sub-resources. The processing function internally connects the input sources and output destinations by reading data from the input source(s) and sending out new outputs to devices identified by the URLs, if any. The last output is stored to allow concatenation of tasks. The getter and setter functions are provided with T-RES as programming APIs to be used in processing functions.

This paper solution approach is a little bit similar with our solution but with many differences to our solution approach and let as see some of the differences.

The first difference and also the limitation of this solution is the overall approach of the system. If the application requires doing the same processing task (e.g., Average) on different sets of sensors and/or sends output to different wireless sensors, multiple task resources needs to be defined and the same function have to be stored in each resource because every task resource stores URLs of input sources and destination outputs the T-RES definition.

The second difference of the T-RES with our solution, the T-RES represents processing tasks as a resource but in our solution we model a flexible dynamic and independent module application

that run anywhere in the wireless senor network nodes i.e. on a sensor node, aggregated sensor or gateway node. We also store input that may arrive from any device and send stored output to any other device after processing. In case of T-Res, the processing function is responsible for getting the inputs from the sensors and sending the output, if there is any output.

In contrast to all the prior approaches our work is different with that of a flexible remote user application (new user application, that compute average value) deployment over a wireless link (required value only) in In-Network-process to the wireless sensor based on CoAP, this results improvement of wireless sensor device accessibility over the network and communication energy consumption.

In order to increase efficiency to the system we can process the data locally at some intermediate nodes within the constrained network to send semi-processed data to next node which may do further aggregation until it reaches to the sink or gateway, so sending all information to the gateway is not required. We use an easy and a flexible remote deployment mechanism using CoAP block-wise to transfer and load the new application module to sensor node via constrained network media.

Summary of Related works

Authors	Title	Approaches	Drawbacks
JaeinJeong and David Culler	Incremental network programming for wireless sensors.[35]	- A reprogramming of the sensor node of the image by using only the incremental code update mechanism which transmits an edit map encoding the differences between old and new program images. - They generate the differences between the two files using a divide-and-conquer dynamic programming approach	- Does not use block level code comparison, and send differences at byte-level granularity.
Panta, R., Khalil, I. and Bagchi, S.	Stream: Low Overhead Wireless Reprogramming for Sensor	- Present a protocol called Stream that mitigates the problem by significantly reducing the size of the program image. - Using multiple code images on a node and	- Installation of images is based on static linking - Memory and energy consumption of the sensor node is not

	Networks.[37]	switching between them, - Stream pre-installs the reprogramming protocol as one image and the application program listen to new code updates as the second image.	considered
Nirmala, M. B., and A. S. Manjunath	Mobile agent based secure code update in wireless sensor networks.[39]	- Developed a protocol and disseminate the code updates in entire networks using mobile agents. - Sensor node should authenticate the mobile agents and check the integrity and confidentially of the code to certain about the origin of the program image.	- The general approach is different with our solution - The agent should always traverse actively in the network which needs more resources.
Alessandrelli, D.; Patracca, M.; Pagano, P	T-Res: Enabling Reconfigurable in-Network Processing in IoT-Based WSNs.[41]	- Present a T-Res programming solution, written in python, which models processing tasks(that internally connects the input sources and output destinations) as resources, sub-resources and make available on each constrained device - The getter and setter functions are provided with T-RES as programming APIs to be used in processing functions.	- If the application requires doing the same processing task on different sets of sensors and/or sends output to different sensors, multiple task resources needs to be defined

Chapter Four: Proposed Work

4.1 Introduction

As we discussed in the previous chapters and reviewed some related researchers publication works, deployment of application module to a constrained sensor node remotely over a constrained network (from anywhere), this makes the nodes more easily accessible and maintenance up to date overall system, especially constrained sensor nodes that are deployed in limited human accessibility areas i.e. load a new module, upload a program, maintain, keeping up to date software information. As well as it gives more flexible environment to constrained sensor network. To describe our proposed design, *design and develop a new user application module and flexible deployment to a constrained sensor node remotely* in an easy manner, we divided in different phases. First we introduce the general overview of normal constrained network traffic flow in section 4.1. In section 4.2 we discuss the proposed system architecture step by step to achieve the general goal of the proposed solution. Finally in Section 4.3 we use a web based ***URI-QUERY*** to enable the sensors to associate or subscribe with aggregator sensor node so that the sensor nodes can send their generated data to registered aggregator sensor node only, then the aggregator sensor node process the input data with the module already deployed on it.

4.2 Normal Constrained Network Traffic Flow

In this section, we present and compare how packets are routed in the original constrained network paradigm and the proposed constrained network paradigm of the Low Power and Lossy Network. Both topologies shows the sensor nodes, Zolertia Z1 mote sensor node is used, that are organized in multi-hop route based tree to illustrate the different physical parameters of the constrained network and route the reading to gateway then to the so-called base station.

In original constrained network traffic scenario, each node will send its sensing information value to the next aggregate sensor node and the next aggregated sensor node to the next one… just by looking the packet up to network layer only, no further processing is used then it forward to the gateway finally the packet reaches to base station. [42] By doing such route if we have "n"$number\ of\ sensor\ node$ in the constrained network, "$n * number\ of\ hops$" number of transmission packet will be routed in the constrained network to reach from initial point of the

sensor node to the base station Figure 4.1. This scenario consumes bandwidth as well as communication energy of each constrained network.

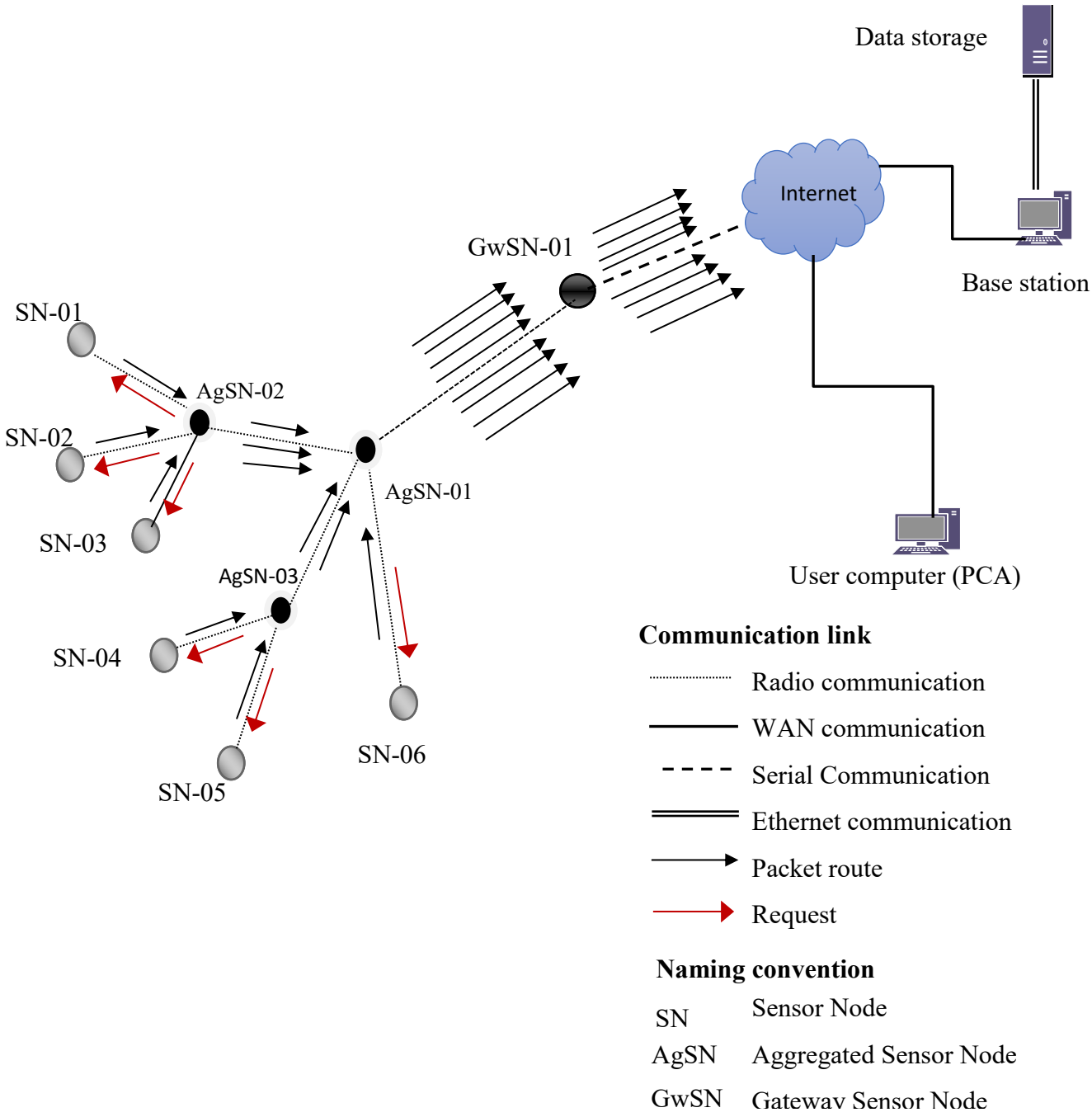

Figure 4. 1: Existing traffic flow

4.3 Proposed Solution

As described in the previous sections, this approach involves a lot of data communication within the constrained network. In many applications, such as warehouse monitoring, we are not interested in every detail of the data. Rather, we wanted to know the general information. In such applications, exchanging all these data between all nodes and the sink or the gateway is inefficient as it involves too much traffic which leads to wastage of already scarce power. In order to increase efficiency to the system we can process the data locally within the constrained network send semi-processed data to the sink, so sending all information to the gateway is not required. Hence, we can do some processing at some intermediate nodes and send the semi-processed data to the next hop until it reaches the sink or the gateway.

Our solution starts by selecting nodes that will host the application module and concludes after the nodes are associated to exchange data (raw or semi-processed) among themselves. Figure 4.2 illustrates the general steps involved in both approaches. The steps involved in achieving this are discussed in detail below.

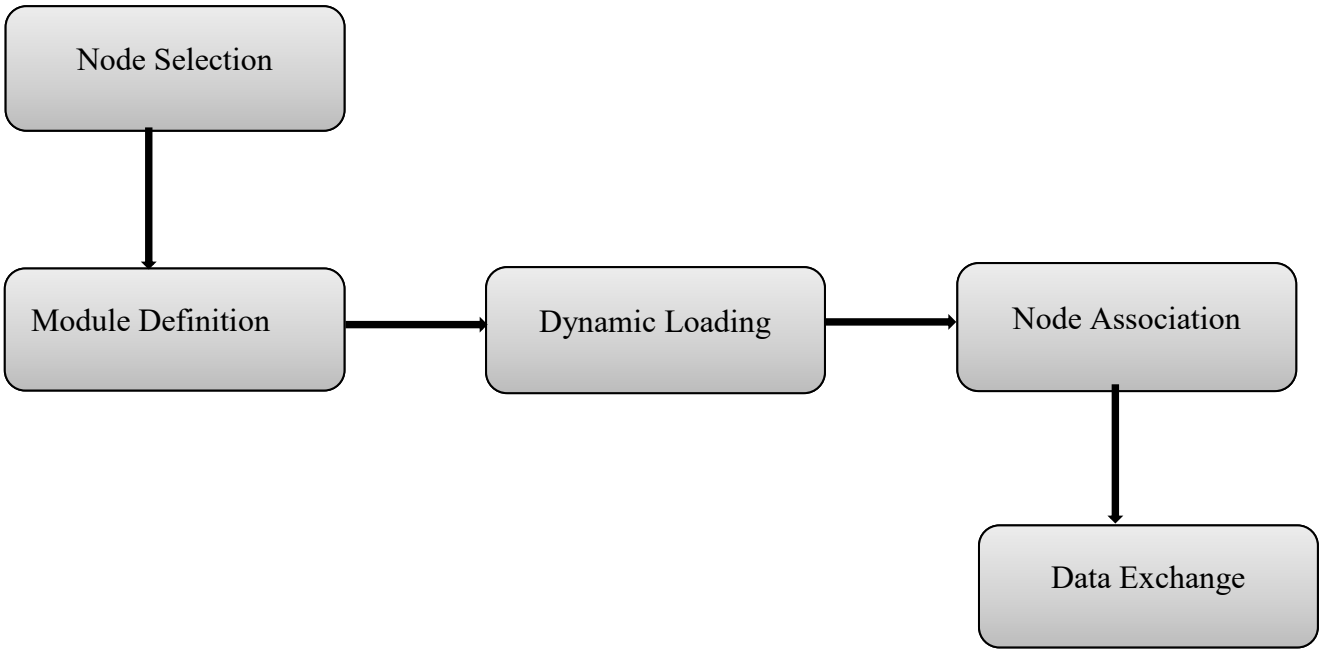

Figure 4. 2: General Steps of the proposed solution

4.3.1 Step 1 – Node Selection

The first step involved in both approaches is selecting nodes that will host the processing module. This step is crucial in order to achieve the most efficient energy consumption reduction. If the right nodes are selected to process the data, the number of data exchange to reach them will be minimal. Hence, the total energy consumption will also be minimal. But, selecting the right nodes is an optimization problem and depends on various factors (such as topology and routing tree). Selecting optimal nodes is beyond the scope of this research work. For the sake of simplicity, we consider a random topology and select nodes by visually inspecting the topology.

4.3.2 Step 2 – Application Module Definition

Application module definition is module that will run on the selected nodes. The application module may vary depending on the requirement of the IoT application. Due to the memory constraint and limited processing capacity of the nodes, complex processing activities cannot be performed. Rather, simple pre-processing tasks such as averaging, finding minimum or maximum values or counting occurrence of events is considered to be performed by the constrained nodes. As an example, we define an averaging module that collects input data from multiple sensor nodes and produce an output that can be communicated further. Definition of such a module is given below.

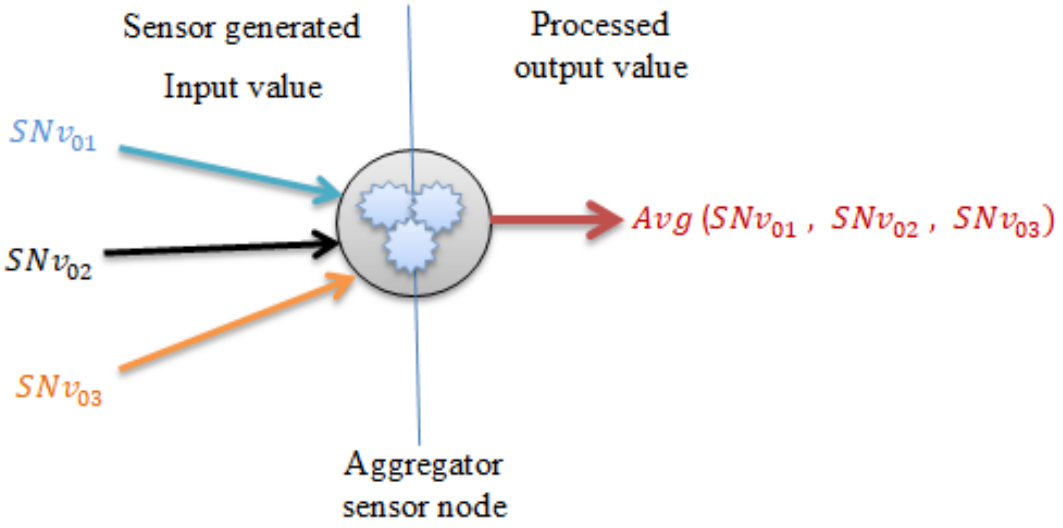

Figure 4. 3: Proposed Module Architecture

Suppose AgSN-02 module is designated to compute the average values which is generated and sent from sensor nodes SNv-01, SNv-02 and SNv-03 with values, a, b and c, respectively. Let us assume calculated result value is $\boldsymbol{SNavg}$.

$$\boldsymbol{SNavg} = avg(a, b, c)$$

AgSN-01 follow the same procedure by computing the average of average sensing information values which is sent from aggregate sensor nodes or just sensing values only, if the node is not an aggregate sensor node and let as assume $\boldsymbol{Avg}\ (\boldsymbol{d}, \boldsymbol{e})$ for AgSN-03 then the final calculated value is going to be $\boldsymbol{FINavg}$.

$$\boldsymbol{FINavg} = \boldsymbol{Avg}\ (\boldsymbol{Avg}\ (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}), \boldsymbol{Avg}\ (\boldsymbol{d}, \boldsymbol{e}), \boldsymbol{f})$$

Any small aggregation function can be defined in a similar fashion.

The proposed designed of the application module algorithm task process are demonstrated below as shown in Figure 4.4

Average value processing

Step1: *get generated data value of SNv-01 as* $\boldsymbol{a}$

Step 2: *get generated data value of SNv-02 as* $\boldsymbol{b}$

Step 3: *get generated data value of SNv-03 as* $\boldsymbol{c}$

Step 4: *compute the* $\boldsymbol{Average}(\boldsymbol{a},\ \boldsymbol{b},\ \boldsymbol{c})$

Step 5: *send* $\boldsymbol{Average}$ *value to next aggregate node*

Figure 4. 4: Module application algorithm

4.3.3 Step 3 – Remote Deployment

Once the application module is defined and compiled for the right platform, it will be loaded into the selected node. There are two options to load the module. The first method is statically linking the module to the rest of the executable file before uploading it. This entails flashing the entire code on a node. This is very inflexible and, in larger networks, it is practically impossible to achieve. The second method, which is proposed in this paper, is flexible remote deployment. We use CoAP Block-wise transfer to transfer the code to the selected node and the nodes operating system's dynamic loading functionality to load the code into memory and link it to the existing modules dynamically.

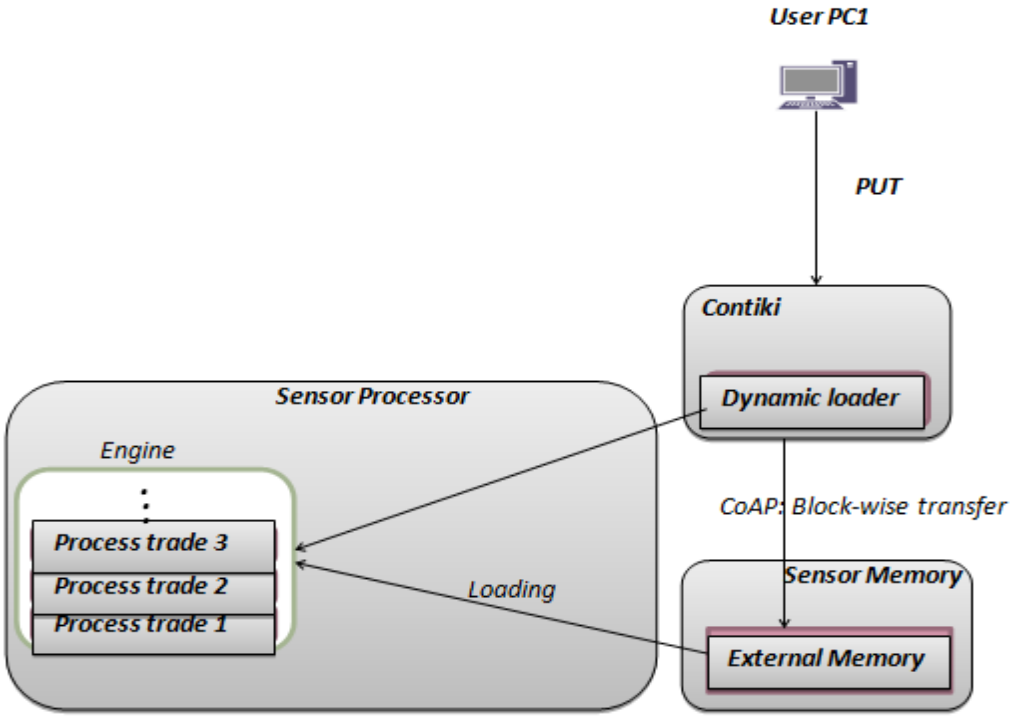

Figure 4. 5: Dynamic User Application Module Deployment Architecture

4.3.4 Step 4 – Node Association

Once the application module is deployed on selected nodes, the next step will be associating the sensor nodes with the aggregator nodes (the nodes which host the dynamic application module). There are two approaches that can be used to achieve this – Periodic Polling or CoAP Observe based. If periodic polling is in effect, the Figure 4.6: Traffic Flow with Periodic Polling is used

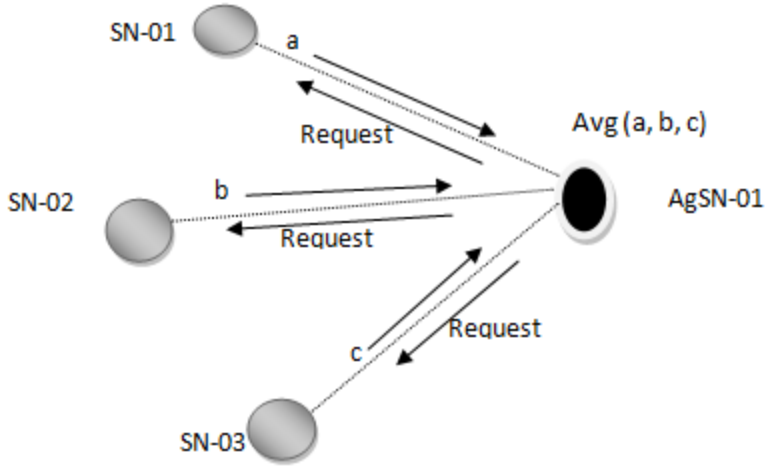

Figure 4. 6: Traffic Flow with Periodic Polling

Aggregator node periodically collects data from specific nodes and processes it. The result will also be sent to the next aggregator or the final destination through the same method and described in Figure 4.7.

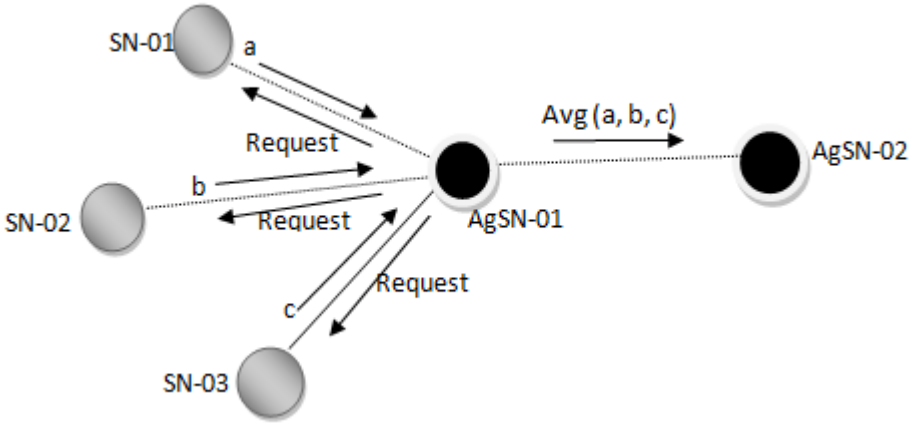

Figure 4.7: Traffic Flow within Aggregator sensor nodes

But if CoAP Observe is used, we send URI-Queries from a third party device (e.g. Smartphone) to the sensor nodes that inform the nodes to subscribe the aggregator node as an observer. The query may specify IP Address of the aggregator, port number and resource URI. Upon receipt of

the query, the sensor node registers the aggregator as an observer and sends notifications whenever the resource state changes only (Figure 4.8).

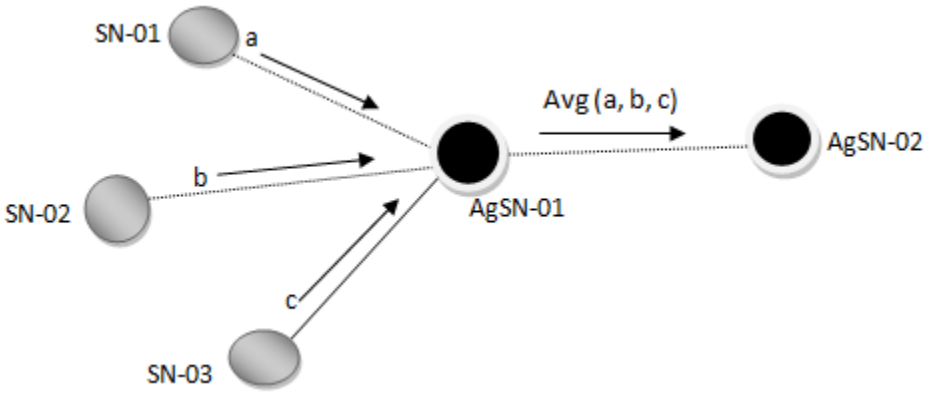

Figure 4.8: Traffic Flow with observe

The pseudo code demonstrates to enable the sensor node to associate with aggregator sensor node.

Node Association Pseudo code

Input: *Observe request*

Process

1: *receive observer request*

2: *IF URI request exist*

3: *Observe ip= URI-QUERY(host)*

4: *Observe uri= URI-QUERY(res)*

5: *Register(observer_ip, observer_uri, METHOD_PUT)*

6: *ELSE*

7: *Register(src_ip)*

8: *ENDIF*

Output: *send sensing value/notification to* AgSN

4.3.5 Step 5 – Data Exchange

Finally, after every application is remotely deployed and dynamically loaded, associations between nodes is made, data exchange begins. The data exchange mechanism varies a bit depending on the approach selected. Periodic polling requires the aggregator node to take the lead in pulling the data from each associated nodes periodically. This is achieved by sending GET requests periodically to the nodes. But for the CoAP Observe based approach, the sensor nodes will be responsible for notifying the aggregator node. Every time the aggregator gets data from the sensors, it initiates the application module to process the data and generate new result.

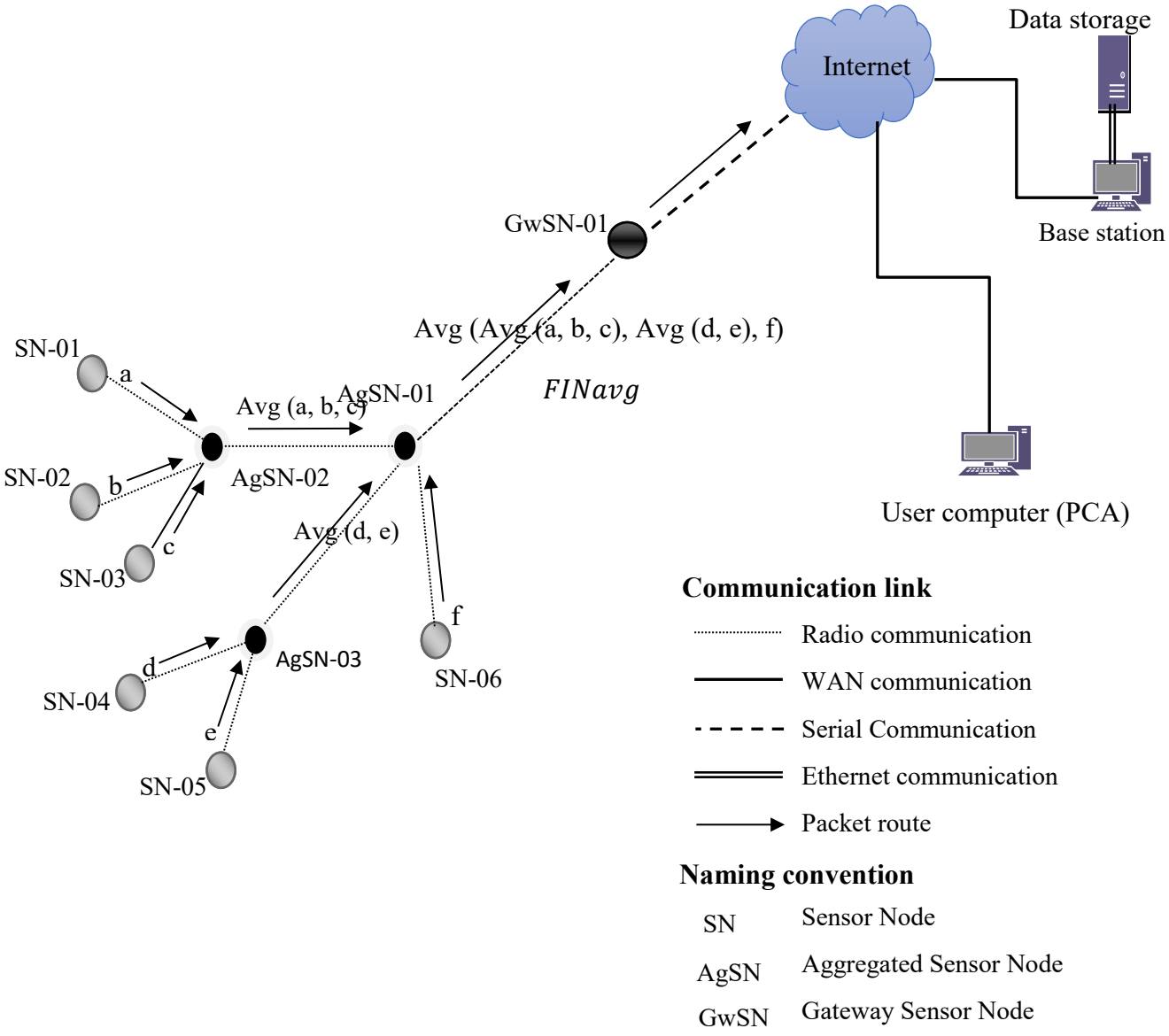

Figure 4. 9: Proposed WSN Traffic Flow

4.4 Node Association through Queries

Now we have all resources we need, i.e. the application module designed in step 2, and is already deployed dynamically from work station see in step 3, to aggregate generated data and send it to the next hope constrained sensor nodes, but still all the resources are idle and the processes are not yet stated or initiated. To do so we use a web based ***URI-Query*** that observer changes (if there is any) to initiate the sensor node to associated with an aggregator node. In order to achieve this, we implemented a modified version of the standard CoAP Observe operation.

The normal scenario of sensor node and next hop sensor communication is using sensor registers the sender as an observer and sends notifications to it whenever a new reading emerges, after the device knows the sensor node then the device (aggregator or the application loaded sensor) sends a GET request to a sensor node to be notified whenever there is a change by including the *Observe* option in the request.

In our proposed solution, we modified the above/original observe function to allow third-party devices to send observation request to a sensor node and request it to register the aggregator as an observer rather than itself. This solution involves associating a resource on a sensor to a resource on the aggregator node. In order to achieve this, the third-party has to include, as URI-QUERY, the aggregator's IP address and the URI_PATH of the resource that accepts the data from the sensor in the observation request. Upon receiving this, the sensor node registers the aggregator as an observer. Whenever there are changes, the sensor node will notify to the aggregator by sending new PUT requests to the resource indicated by the URI-PATH on the aggregator. Figure 4.10 shows the pseudo code or syntax demonstration for the observe request:

```
GET [Sensor-IP]
URI-PATH uri
URI-QUERY host=[AggregatorIP]&res=aggrigator_uri
OBSERVE
```

Figure 4. 10: QUERY Pseudo code for SN-0i value

Chapter Five: Implementation and Results

5.1 Introduction

In this section, we present the implementation of the proposed system and evaluation results. We will show the functional and performance evaluation results.

We performed a simulation experiment and evaluate using different metrics. To accomplish this we followed different procedures: first, we defined the simulation setup where it encompass defining the arrangement of constrained sensor nodes in the network, identifying the type of constrained sensor nodes and define the nodes type in section 5.1 in the network, develop an application module, deploy it remotely, define a web based URI-QUERY that associate the sensor node to aggregator sensor node. Second, we conducted computer simulation by deploying different software tools that are developed for Low Power and Lossy Network application. Lastly, we determined the evaluation metrics that help us to observe and evaluate the flexibility of the system, easy of management, consumption of the communication energy of the proposed system implementation in Section 5.1 and then we described our experimental analysis in Section 5.2.

5.2 Implementation

The proposed solution was implemented on Contiki OS version 2.7. Contiki is an open source embedded operating system that can be installed on smart objects such as sensor and actuator nodes. Since we lack real sensors and wireless test bed, we used Cooja Simulator to test the performance of our solution. The simulated nodes are Z1 motes with limited memory and processing capacity. Z1 nodes are selected because of this limitation to show that in-network processing is still possible with constrained nodes. The official CoAP implementation of Contiki, Erbium was extensively used in the proposed solution.

5.2.1 Zolertia Z1 Mote

Different sensor nodes are designed and manufactured by different companies [17] but because of its many advantages which is described next we choose Zolertia Z1 mote sensor node type, produced by a Spanish company named Zolertia, in our thesis.

The Z1 is a low power wireless module compliant with IEEE 802.15.4 and Zigbee protocols. Its core architecture is based upon the MSP430+CC2420 family of microcontrollers and radio transceivers by Texas Instruments. However, the microcontroller unit (MCU) that Z1 uses is the MSP430F2xxx instead of the MSP430F1xxx, as is customary among other motes, like Crossbow's TelosB, Moteiv'sTmote, and alike. The inner changes between F2xxx and F1xxx devices lead to the subtle differences between Z1 and other F1xxx devices, and these differences in turn result in that some Contiki functions available for Tmote sensor nodes are not portable to the Z1 sensor nodes. Z1 has two built-in sensors, one accelerometer sensor and one temperature sensor. The ADXL345 is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I2C digital interface. The TMP102 is ideal for extended temperature measurement in a variety of communication, computer, consumer, environmental, industrial, and instrumentation applications. The device is specified for operation over a temperature range of - 40°C to + 125°C. The more information and documentation about these two sensors can be found in [50]. In addition to the built-in sensors, Z1 also supports up to four external sensors.

- ✓ **Zolertia Z1 Microcontroller**

A Z1 sensor node, made by Texas Instruments [18] is equipped with the low power microcontroller ***MSP430F2617***, which features a powerful 16-bit RISC CPU @16MHz clock speed, built-in clock factory calibration, 8KB RAM and a 92KB Flash memory.

- ✓ **Zolertia Z1 Transceiver**

Zolertia Z1 Mote also includes a ***CC2420 transceiver*** from Texas Instruments, which is IEEE 802.15.4 compliant and operates at 2.4GHz with an effective data rate of 250Kbps. Z1 hardware selection guarantees the maximum efficiency and robustness with low energy cost. The CC2420 is a true single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver designed for low-power and low-voltage wireless applications. CC2420 includes a digital direct sequence spread spectrum baseband modem providing a spreading gain of 9 dB and an effective data rate of 250 kbps.

5.2.2 Copper and Erbium

Copper is a CoAP protocol handler for Mozilla Firefox, used for browsing and book marking of CoAP URIs and Interaction with resource like REST Client or Poster. It treats tiny devices like normal Restful Web services. It is designed for unconstrained environments as Cf. Its ability to render a number of different content types such as JSON or the CoRE Link Format makes it a useful testing tool for application as well as protocol development.

Erbium is a low-power REST Engine for Contiki that was developed together with scientific computing and makes low-power systems to communicate efficiently and easily with the Internet. Therefore, this implementation is specialized for constrained environments as it is designed to run on small amounts of memory and low-power Central Processing Units or Microcontroller Units.

5.2.3 Node Association

As discussed in the previous chapter, one of the steps involved in the proposed solution is **node association,** a phase were a sensor node is associated with an aggregator node. The solution implemented here is the CoAP Observe based solution. This solution involves associating a resource on a sensor to a resource on the aggregator node. In order to achieve this, we implemented a modified version of the standard CoAP Observe operation.

As discussed above, the normal CoAP observe specification states that a device sends a GET request to a sensor node to be notified of every state change of the sensor by including the *Observe* option in the request. Upon receipt, the sensor registers the sender as an observer and sends notifications to it whenever a new reading emerges.

In the modified version, we modified the normal observe function to allow third-party devices to send observation request to a sensor node and request it to register the aggregator as an observer rather than itself. In order to achieve this, the third-party has to include, as URI-QUERY, the aggregator's IP address and the URI_PATH of the resource that accepts the data from the sensor in the observation request. Upon receiving this, the sensor node registers the aggregator as an observer. Changes at the sensor will be notified to the aggregator by sending new PUT requests to the resource indicated by the URI-PATH on the aggregator.

APPENDIX A: code script shows the modification of observer function (***er-coap-13-oberserving.c***) done on the erbium.

5.2.4 In-Network Processing

In-network processing or aggregation is done on aggregator nodes which may or may not be sensor nodes. The code on these nodes will have a resource defined for each input it receives from a sensor node. As indicated above, the sensor node will be programmed to send PUT requests to this resource on the aggregator node whenever the sensor reading changes. Whenever these resource state changes, the processing module will be initiated in order to process the data and produce the result. The result may then be sent to the next aggregator sensor node or the gateway or the cloud for further processing and storage.

APPENDIX B: shows a code script in erbium modification done on ***er-coap-dyn-module.c*** resource file.

5.2.5 Remote Deployment

Remote deployment and loading process starts with compiling the application module with the appropriate flags. Figure 5.4 simulation screenshot shows a compilation process of the deployment and loading. The Contiki make rule compiles the program as a Contiki application and strips off unnecessary components from the file. This modification is done at er-dynamic-loader resource on erbium.

The ***er-dynamic-loader.c*** code script modification done on erbium has in **APPENDIX C.**

5.3 Experiment Setup

Experiments are conducted on Lenovo Laptop-E51-80 computer which runs Ubuntu14.04 platform with the Long term support (LTS). The laptop has 8GB RAM and Intel® Core ™ i7-6500U CPU@2.50GHz 2.50GHz processor. For the purpose our experiment we installed Contiki OS 2.7 with Cooja constrained network simulation software, which is described in section below in detail, on Ubuntu 16.04 LTS laptop computer.

5.3.1 COOJA

COOJA, which is used for simulation and emulation purposes in this thesis, is a simulator for the Contiki sensor node operating system from small sensor nodes to a very larger sensor nodes network simulation. COOJA allows for simultaneous simulation at different level of the system. [32].

- Network level,
- Operating system level, and
- Machine code instruction set level.

Cooja is a tool for Contiki development as it allows constrained sensor nodes to be emulated at hardware level in Instant Contiki OS with the following different performance behaviors,

- slower but allows precise inspection of the system behavior,
- less detailed level, which is faster inspection of the system behavior

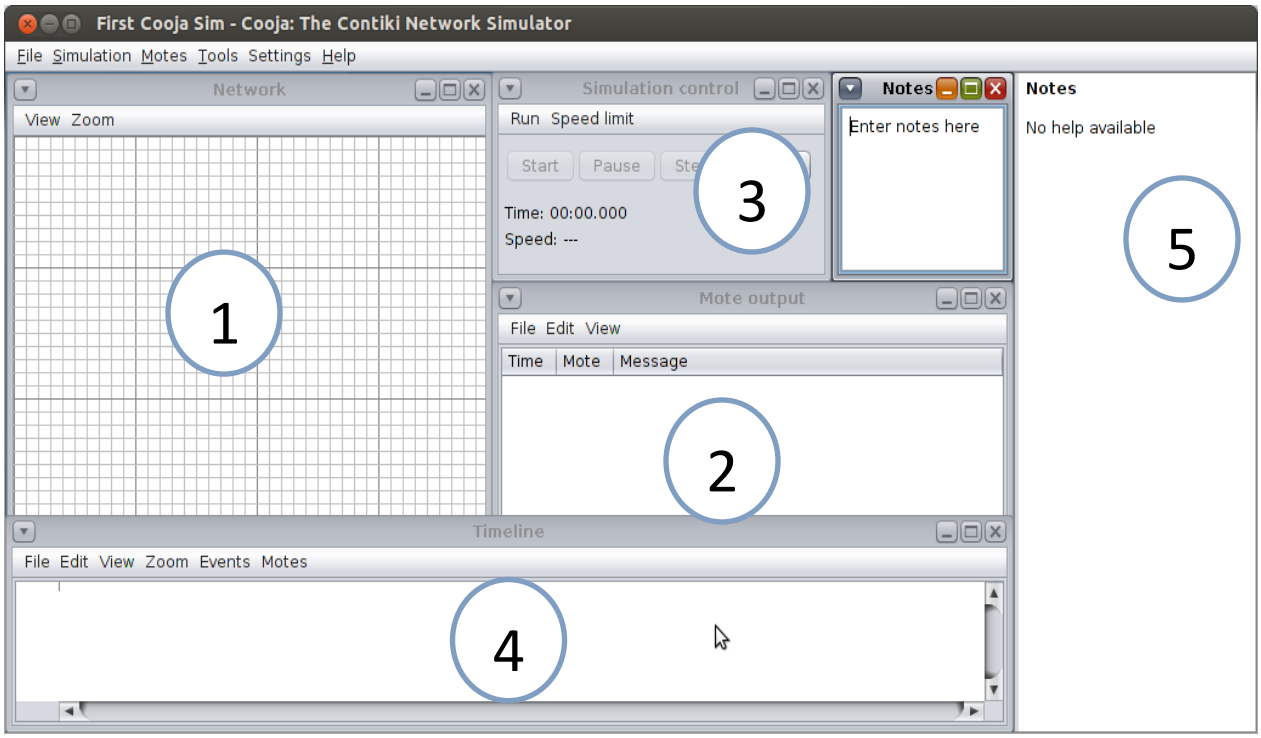

Figure 5.1: Cooja Contiki Network Simulator Interface

Here we briefly describe the functionalities of each tool:

1) **Network** - Shows the location of each node in the network. Can be used to visualize the status of each node, including LEDs, mote IDs, addresses, of outputs, etc. Initially this window is empty and we need to populate it with our sensors.
2) **Mote output** - Shows all output of serial interface of the nodes. It is possible to enable one window of Mote output for each node in the simulation.

3) **Simulation Control** - This panel is used to Start, Pause, Reload or execute Steps of the simulation. It shows the time of execution and the speed of simulation. It means that we can run the events several times faster than it would take in real-time execution.
4) **Timeline** - Simulation timeline where messages and events such as channel change, LEDs change, log outputs, etc. are shown
5) **Notes** - This is a simple notepad for taking notes about the simulation.

5.3.2 Simulation Topology

For our experiment, we use four motes (three client constrained node two server constrained node, one for aggregated sensor node and one for a gateway). From these three clients constrained nodes are connected to aggregated sensor node and the aggregated sensor connected to the gateway and the gateway to workstation via Internet forming a hierarchal constrained network, or Low Power and Lossy Network (LLN). The client nodes are used to generate data value, sensing temperature from the environment and send it the value to the next hope aggregator or server node which is an application module deployed on it and the aggregator sensor node compute average value and again send the result value for next hope or gateway node which may do further aggregation. We used our Lenovo E51 laptop with Ubuntu 14 operating system as a work station. To accomplish the above tasks, we implemented the following, Contiki OS: for the network simulation environment, COOJA for simulation and emulation. Finally in last sections we describe the different scenarios with their respective simulation experiment and evaluation result.

We take different topologies scenarios by varying number of hopes to analysis the functional and performance evaluation differences between original constrained network traffic with our designed solution, the following screenshots are taken from the topology simulation. See Figure 5.2.

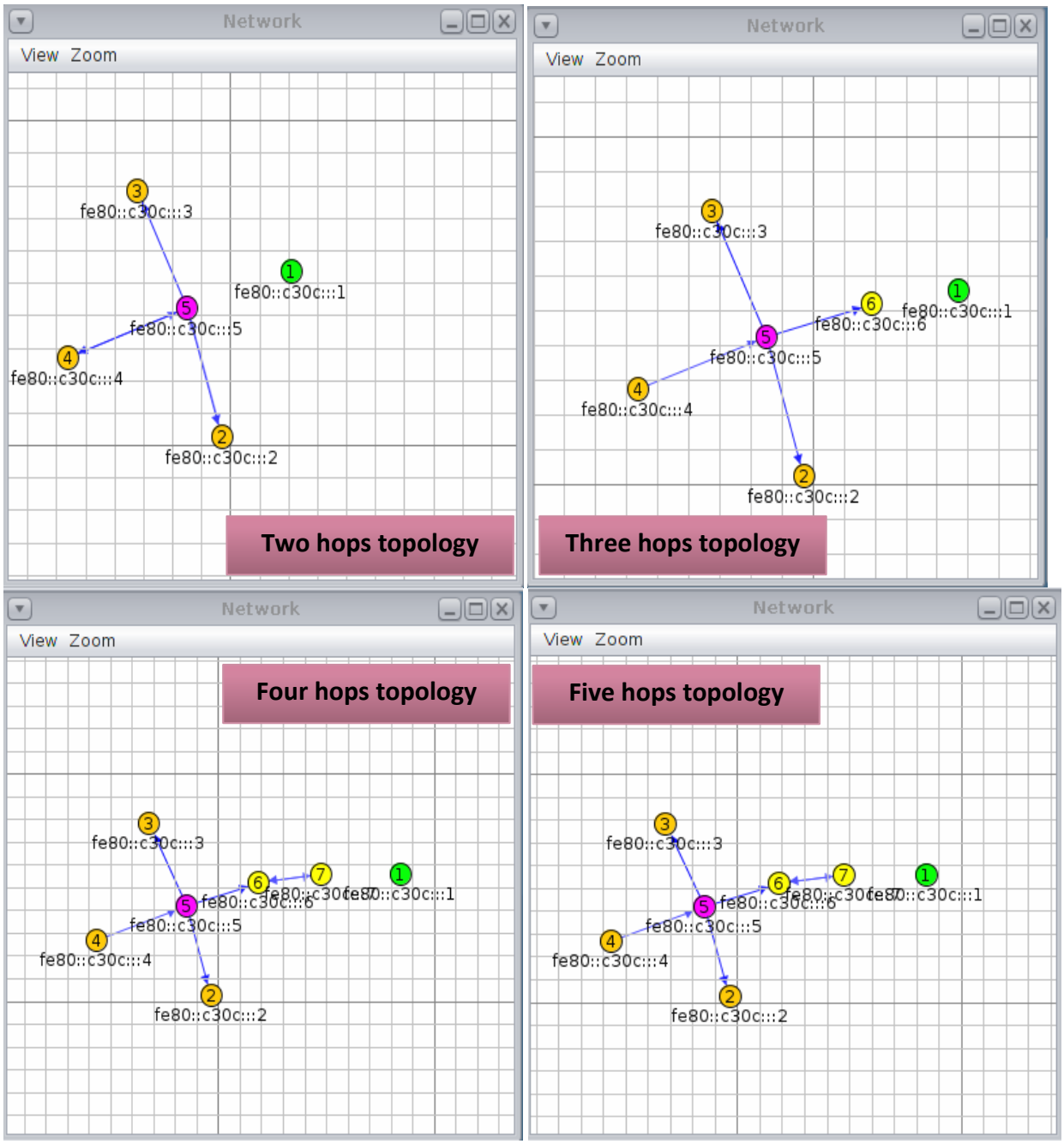

Figure 5. 2: Different hop topology scenarios

5.3.3 Data Source

In real world, sensor nodes can read an environment data but this is not possible in simulation software. In order to emulate sensor data, we generated 100 random numbers and stored it in an array. The handler function is invoked every 5 seconds to read the data from the array. The index of the array is incremented in order to get new readings during each iteration.

The erbium data source function code namely ***er-sensor.c*** is shown in **APPENDIX D**.

5.4 Evaluation

We compared the number of packets required to complete the communication, and energy consumption of the proposed solution against a default constrained network solution. All tests were run 10 times for each topology and the averages are taken for comparison.

5.4.1 Functional Evaluation

❖ Node association from Copper

Sending a web based URI-QUERY from third party or copper to end sensor nodes and enabling the sensor node to register to the aggregator sensor node using the IP address, port address, URI path. The screenshot below demonstrate the implementation.

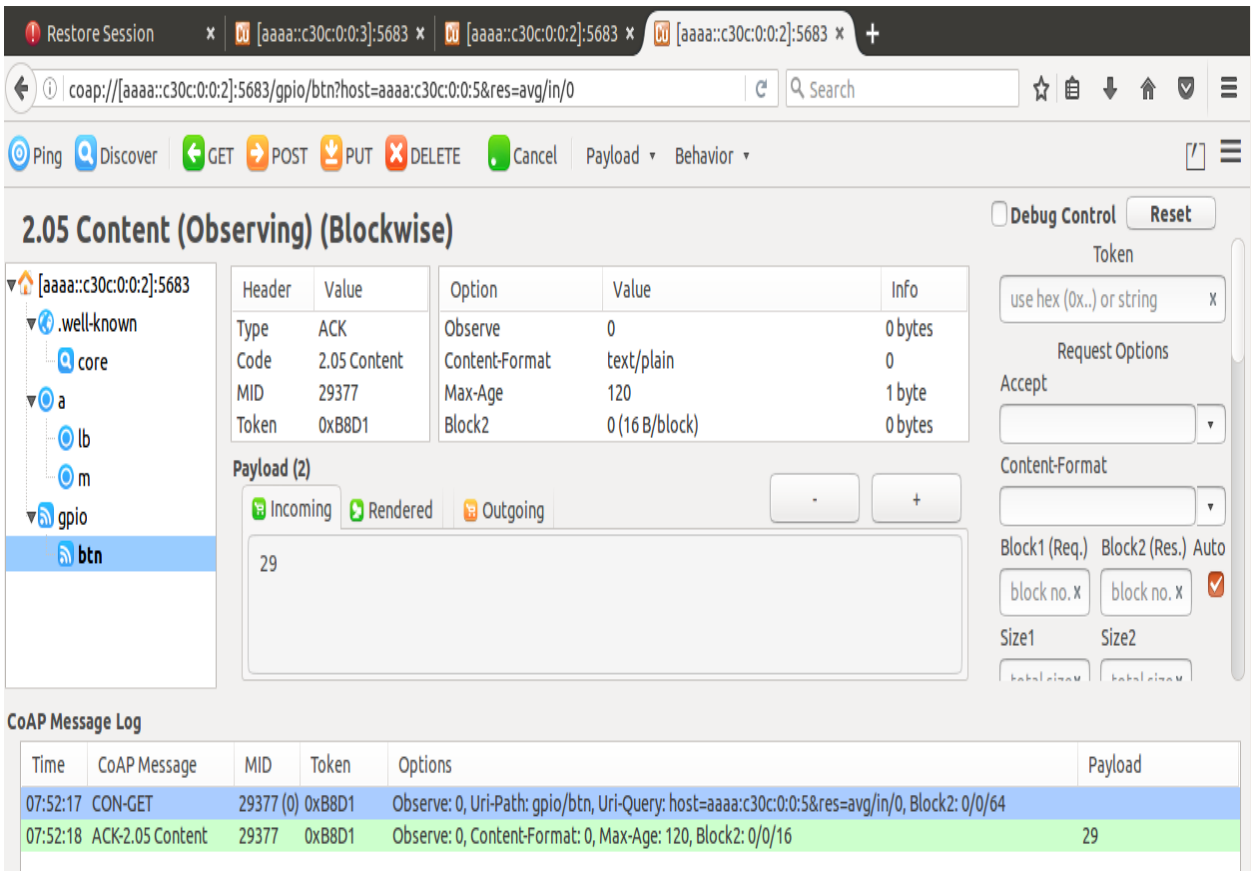

Figure 5. 3: Node Association

❖ Remote Deployment

The screenshot below depicts an easy and flexible remote deployment of application module (a module that compute average temperature value) using CoAP block-wise transfer protocol. The simulation below shows data transferring from base station to aggregator sensor using maximum block size is 64B/blk and start loading the application module dynamically.

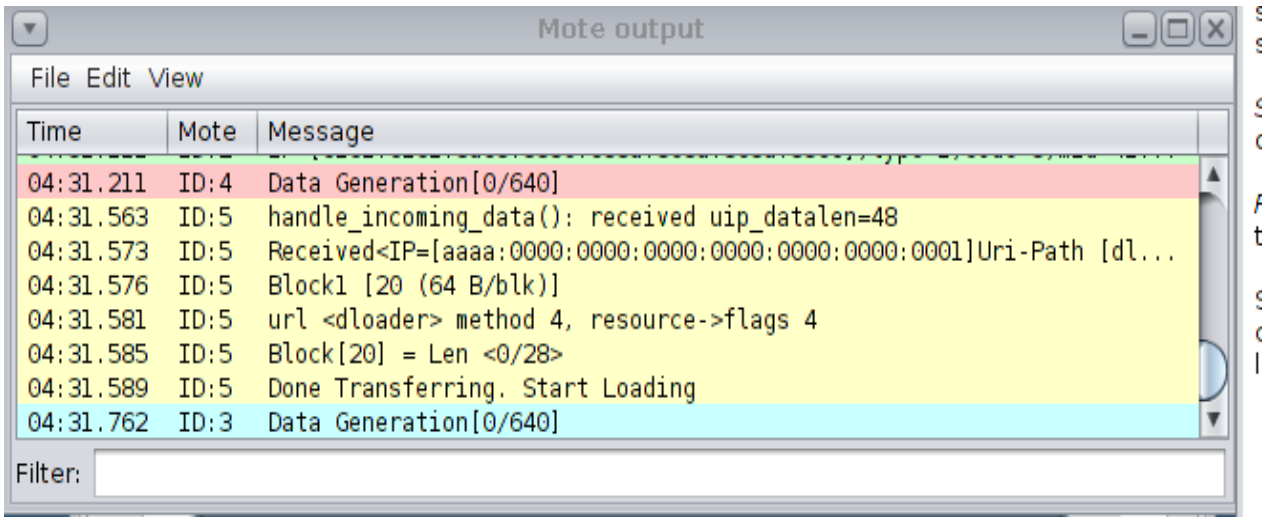

Figure 5. 4: CoAP Block-Wise based Remote Deployment and Loading a Module

❖ Aggregation

After remote deployment of the module done on the aggregator sensor node, the sensor nodes send their generated data based on the URI-QUERY information given and the aggregator sensor node receive each sensor nodes value and aggregate to send to next hope aggregator sensor node. Figure 5.5: shows aggregation simulation.

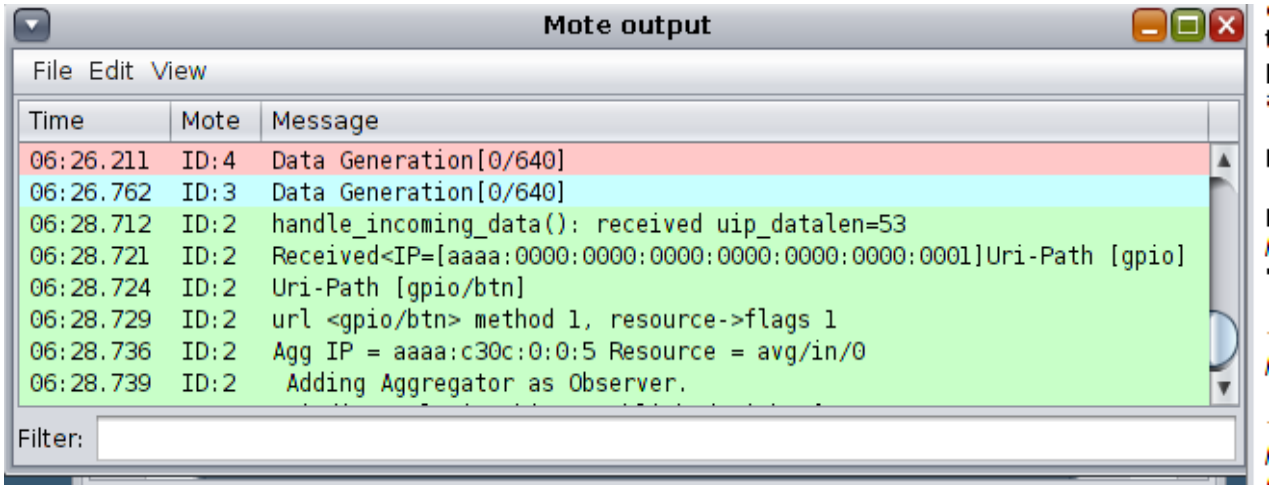

Figure 5.5: Aggregation

5.4.2 Performance Evaluation

5.4.2.1 Number of Packets

We compared the number of packet transaction in the constrained network by varying the number of hopes from sensor nodes to base station against the number of packet Figure 5.6 displays in detail in the graph below.

Using a simple average, suppose AgSN-02,the module is designated to compute the averages value which is generated from sensor nodes SN-01, SN-02 and SN-03 with values, a, b and c, respectively. The average is computed and resulted as $\boldsymbol{SNavg}$.

$$\boldsymbol{SNavg} = avg(a, b, c)$$

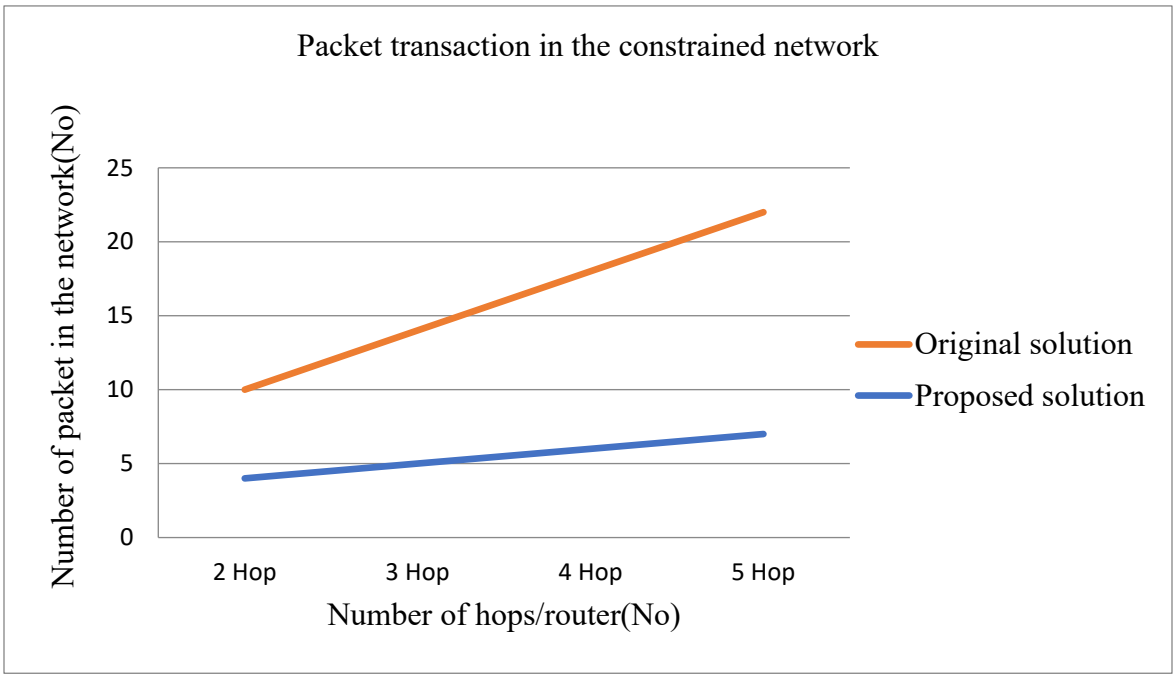

Figure 5. 6: Network packet transaction

5.4.2.2 Energy Consumption

The energy consumption is the sum of used energy of all the nodes in the network, where the used energy of node is the sum of the energy used for communication, including transmitting, receiving, and idling. In above section, number of packet section, we have seen that the number of packet is decreased as we add more number of hops in the network. Assuming each transmission consumes an energy unit, the total energy consumption is equivalent to the total number of packets sent in the network, and energy is optimized in our designed solution.

❖ Mathematical Evaluation

General mathematical formula to get energy consumption for the overall communication energy of the constrained network is by adding the energy consumption of the sensor nodes times sensor nodes in constrained network plus aggregator sensor node energy consumption times the number of aggregator nodes plus number of routers times energy consumption the routers. The equation is illustrated below.

$$E_{Total} = n * E_{sn} + E_{AgSN} + m * E_R$$

Where

E_{Total}	Total communication energy of the constrained network
E_{sn}	Energy consumption of senor nodes in the constrained network
E_{AgSN}	Energy consumption of aggregator sensor
n	Number of sensors in the constrained network
m	Number of routers in the constrained network
E_R	Energy consumption of routers in constrained network

Now to compute the energy consumption of a single sensor node the equation will be:

$$E_{sn} = E_{cpu} + E_{cpu-lpm} + E_{Tx} + E_{Rx} + E_{idle-radio}$$

Where

E_{cpu}	CPU energy consumption
$E_{cpu-lpm}$	CPU energy consumption in low powered mode
E_{Tx}	Energy of data transmission
E_{Rx}	Energy consumption of data receiving
$E_{idle-radio}$	Energy consumption of radio idle

To get E_{sn} , first we have to compute the value for each of the following:

$$E_{cpu} = I_{cpu} * V * t_{cpu}$$

$$E_{cpu-lpm} = I_{lpm} * V * t_{lpm}$$

$$E_{Tx} = I_{Tx} * V * t_{Tx}$$

$$E_{Rx} = I_{Rx} * V * t_{Rx}$$

$$E_{idle-radio} = I_{idle} * V * I_{idle}$$

Finally, the formula to get energy consumption of a single sensor node computed as:

$$E_{sn} = (I_{cpu} * V * t_{cpu}) + (I_{lpm} * V * t_{lpm}) + (I_{Tx} * V * t_{Tx}) + (I_{Rx} * V * t_{Rx}) + (I_{idle} * V * I_{idle})$$

Using the above equation and constant values below in Table 5.1 we can compute the energy values of any sensor node in the constrained network and we can compare communication energy of the overall constrained network.

Table 5. 1: Constant values

No	Notations	Descriptions	Values
1	V	Voltage	3V
2	I_{cpu}	CPU Current	8mA
3	I_{lpm}	Current for Low Power Mode	0.0005mA
4	I_{Tx}	Transmission Current	17.4mA
5	I_{Rx}	Receiving Current	18.8mA
6	I_{idle}	Idle Current	0.426mA
7	T_{total}	Total time	60s
8	N	Number of nodes	Vary
9	M	Number of routers	Vary
10	t_{Tx}	Transmission Time	0.01s/pkt
11	t_{Rx}	Receiving Time	0.01s/pkt
12	t_{cpu} %	CPU Time Percent	10%
13	t_{lpm} %	Time for Low Power Mode	90%
14	t_{cpu}	CPU Time $t_{cpu} = T_{total} * t_{cpu}$	6s
15	t_{lpm}	CPU Low Power Mode $t_{lpm} = T_{total} * t_{lpm}$	54s
16	$Delta\ t_{cpu-lpm}$	CPU Time and Low Power Mode difference	1%
17	$Radio_{idle}$	Radio Idle in Percent	99%
18	$Radio_{active}$	Radio Active in Percent	1%
19	Td	Time gap between packets	5 ms b/n Tx
20	N	Number of Packets	12 pkts/day

By using the above mathematical formula and value inputs we can get the following discussion and results.

Evaluation of communication energy consumption is taken by varying the number of sensor nodes against energy consumption to compare the original constrained network scenario with the proposed solution of the constrained network. As it can be seen in Figure 5.7, Figure 5.9, and Figure 5.11, as the number of sensor nodes increases communication energy consumption of proposed solution is more efficient than the original constrained network. We sampled 3 sensor nodes and 3 different topology scenarios (2 hops, 3 hops, and 4 hops) in all our energy consumption evaluations.

We also evaluated the original and proposed solution of communication energy by comparing energy consumption against data generation interval by varying the time gap between each packet generated by the sensor nodes. The proposed solution indicates the energy consumption is also more efficient than the original constrained network scenario. We take 3 sensor nodes for our mathematical evaluation and three different topology scenarios (for 2 hops, 3 hops, and 4 hops) as previous topology and the detail result and discussion of energy consumption against data generation interval is described in Figure 5.8, Figure 5.10, and Figure 5.12. From all results displayed in the graphs we can conclude that in our proposed solution there is an efficient usage and energy saving compared to the original constrained network.

➢ Results for 2 Hops

The result of Energy consumption against number of sensor nodes for 2 hops is illustrated in Figure 5.7. The data is generated every 5 seconds.

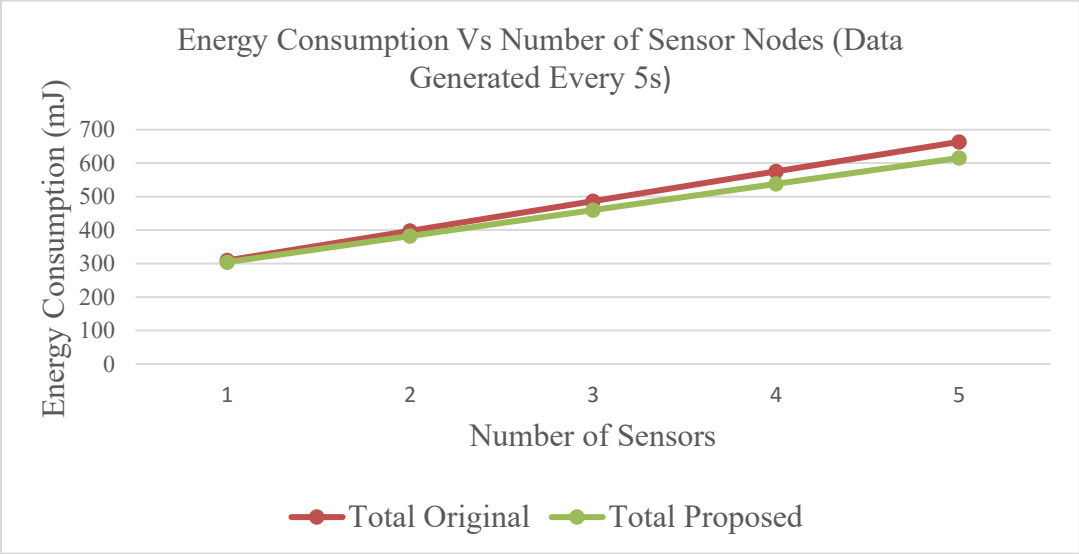

Figure 5. 7: Energy Consumption vs. Number of sensor nodes

We also evaluate the original and proposed solution energy consumption using Energy consumption against data generation interval as shown in Figure 5.8.

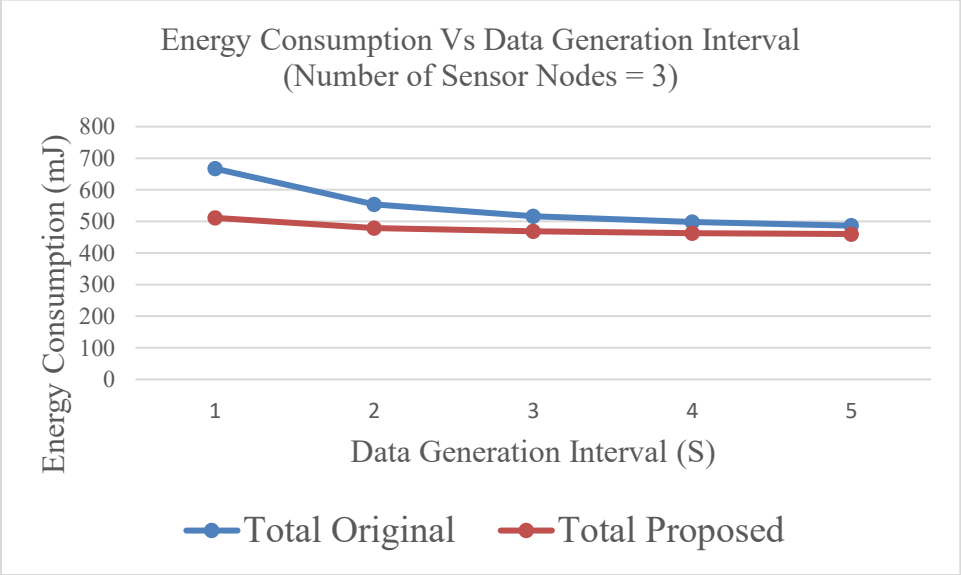

Figure 5. 8: Energy consumption vs. Data generation interval

➢ Results for 3 Hops

The result of energy consumption against number of sensor nodes for 3 hops is illustrated in the Figure 5.9.

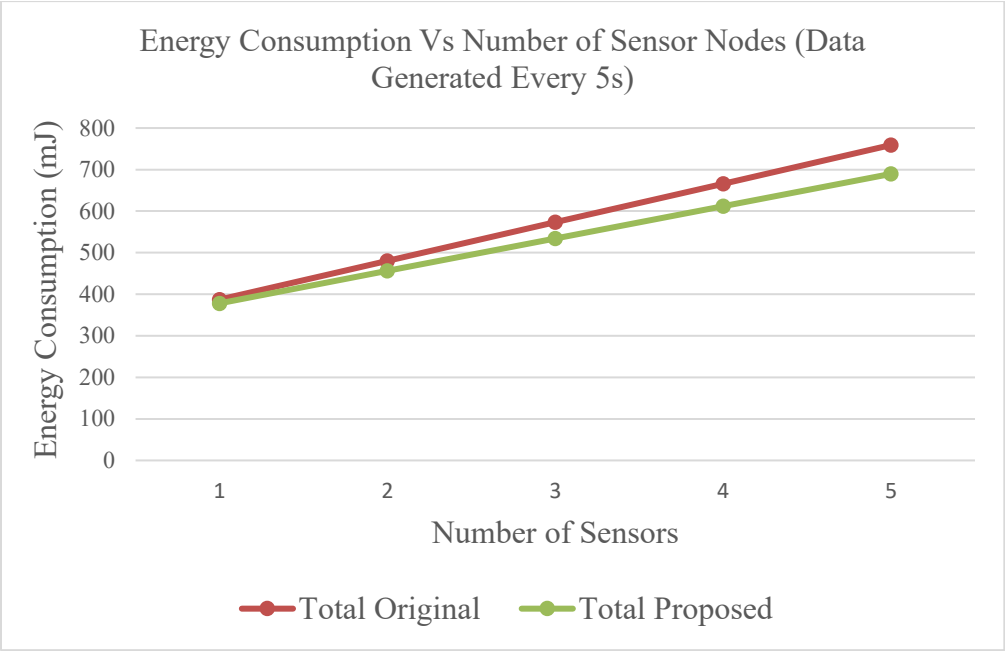

Figure 5. 9: Energy consumption vs. Number of sensor nodes

The result of energy consumption against data generation interval for 3 nodes is illustrated in the Figure 5.10.

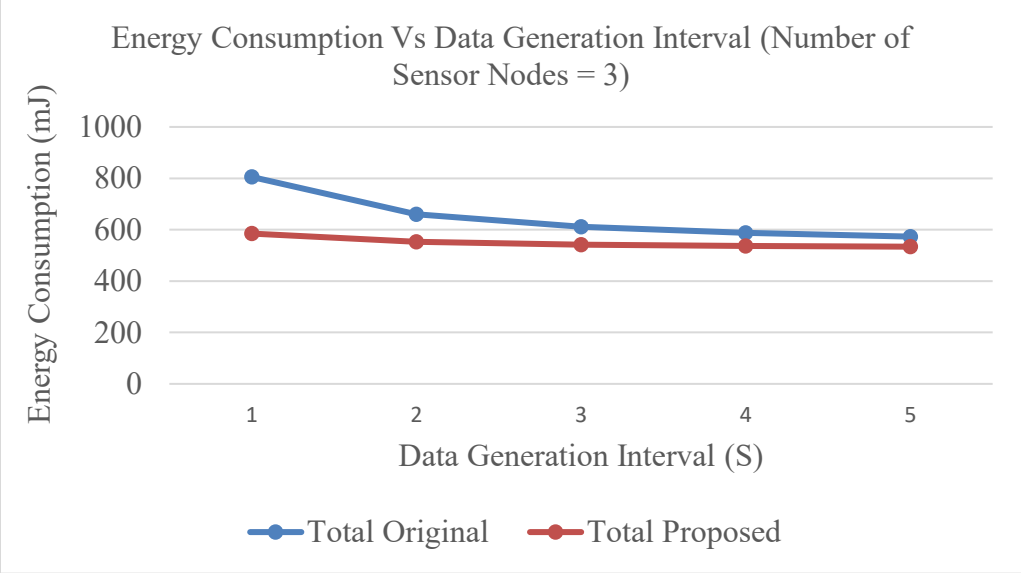

Figure 5.10: Energy consumption vs. Data generation interval

➢ Results for 4 Hops

The result of energy consumption against number of sensor nodes, data is generated in 5 seconds and illustrated in the Figure 5.11.

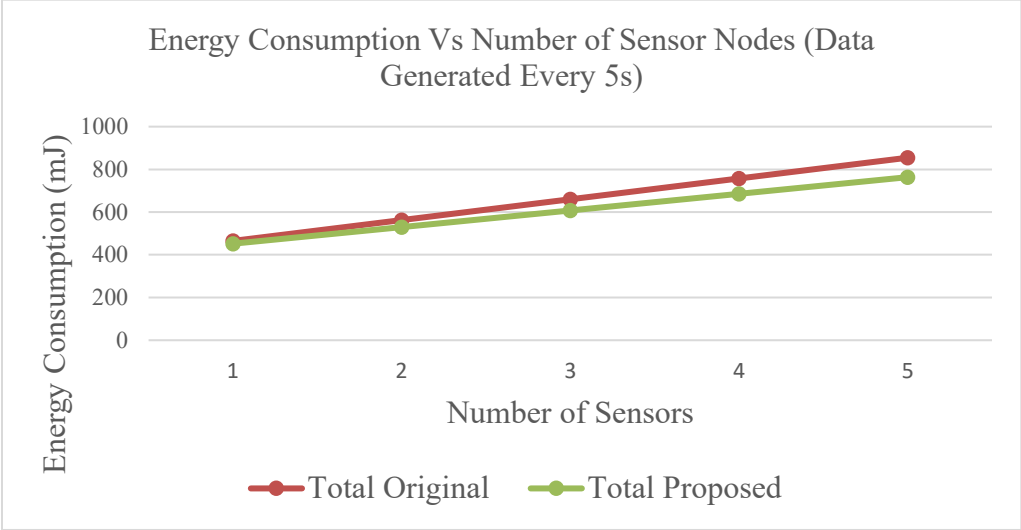

Figure 5. 11: Energy Consumption vs. Number of sensor nodes

The result of energy consumption against data generation interval for 3 nodes is illustrated in the Figure 5.12.

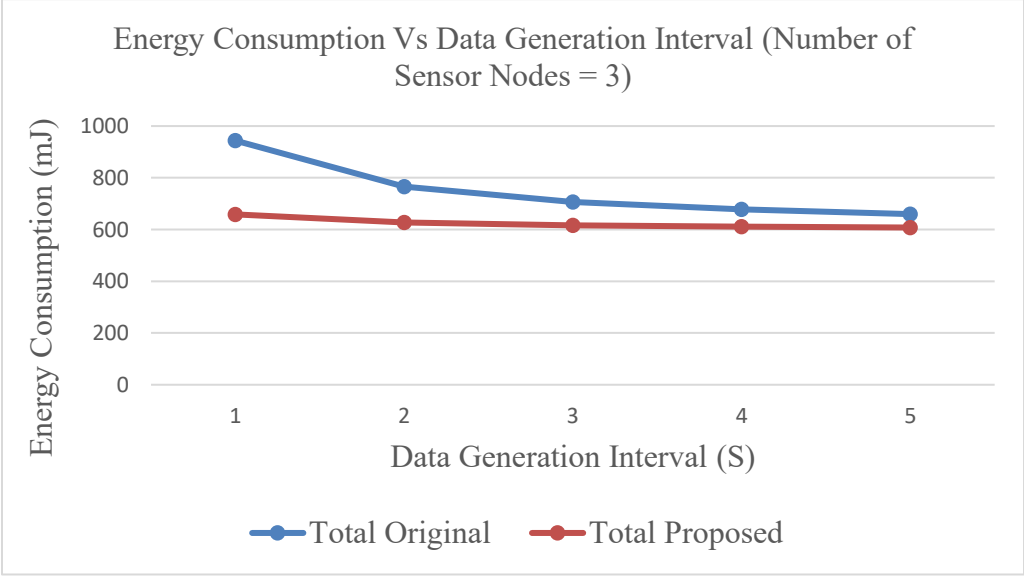

Figure 5. 12: Energy consumption vs. Data Generation interval

❖ Summarized communication Energy Saving (in 60 Seconds)

In general, from the above discussions and results, the energy saving against the number of sensor nodes in constrained network within total time of 60 seconds and with 5 seconds data generation interval can be computed as illustrated in Figure 5.13.

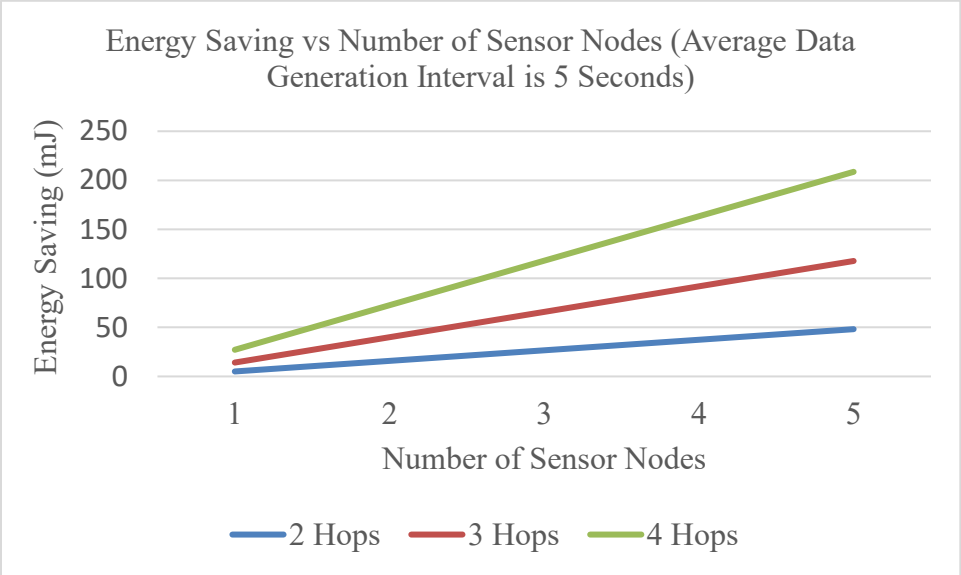

Figure 5.13: Energy saving vs. Number of sensor nodes

Chapter Six: Conclusions and Future Works

6.1 Conclusions

Now a day everything becomes connected using modern digital network and everything becomes programmable to work what function we really need. Billions of sensors and actuators are connected to transform traditional world to digital world, like smart city, smart homes, self-driving cars, and other dangerous areas also digitalized in the newly driven Internet of Things technology, each daily generates billions of huge data and interconnected using LLN. In such constrained sensor and huge number of deployed constrained devices, automation is the most important feature to enable the device accessible, maintenance, update, and upgrade.

The purpose of this research work is to increase efficiency to the system we can process the data locally at some intermediate nodes within the constrained network to send semi-processed data to next node which may do further aggregation until it reaches to the sink or gateway, so sending all information to the gateway is not required. The application module is remotely deployed in an easy and flexible way using CoAP block-wise protocol to such constrained sensors which is interconnected in LLN.

In the final of our work, we able to contribute some solution to constrained network accessibility mechanism to update or upgrade firmware's remotely via a LLN in IoT.

To describe some of our contributions:

- With the complete mechanism, provided an easy and flexible remote application module deployment using CoAP Block transfer protocol mechanism to a constrained node without disturbing the other sensor nodes on the network and limiting unnecessary data transmission over wireless sensor network media by aggregating data generated by multiple sensors.
- In order to evaluate the function and performance of our designed system we used different important metrics for ease of accessibility and flexible deployment and reduction of packet transmission in LLN.

Even though a number of deployment mechanism are proposed by many research works, the methods of their module deployment are different and number of packets are not considered and billions of big data is generated in IoT technology. In our work, we able to contribute some

solution to remote deployment for constrained sensor nodes. Therefore, the result of this research provides its own contribution to the ongoing researches in this domain area.

6.2 Future Works

There are a few aspects that need future discussion and some implementation and operational issues need to be addressed before our designed architecture deployed in real world and to the Internet. The main limitation of this remote deployment mechanism is data compression which is not considered in our work. The more feature or function we add the more code is uploaded to Erbium sensor, the more code is added the sensor memory become loaded and the life time of the constrained device will be shorten even the device may die. To fix these fears, data compression in transferring the payload is the best approach.
Secondly, security is not considered in our proposed work while transferring the data payload to aggregated sensor node. I.e. authentication or some other security measures to make the data as well as the network more secure.

These two issues can be considered as a vulnerability of our research working approach when we are thinking security, so these can be a future work in these domain areas.

References

[1] D. Pompili, T. Melodia, and I. F. Akyildiz, “Deployment analysis inunderwater acoustic wireless sensor networks,” in WUWNet ’06, 2006.

[2] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh and D. Rubenstein, "Energy-efficient computing for wildlife tracking", ACM SIGPLAN Notices, vol. 37, no. 10, p. 96, 2002.

[3] Dunkels, A; Gronvall, B; Voigt, T. Contiki a Lightweight and Flexible Operating System for Tiny Networked Sensors. Proceedings of the 9th Annual IEEE International Conference on Local Computer Networks, Washington, DC, USA, October 2004; pp. 455–462.

[4] Shelby, Zach, Klaus Hartke, and Carsten Bormann. The constrained application protocol (CoAP). No. RFC 7252. 2014.

[5] Bormann, C., and Z. Shelby. Block-Wise Transfers in the Constrained Application Protocol (CoAP). No. RFC 7959. 2016.

[6] Bause, Falko. "Queueing Petri Nets-A formalism for the combined qualitative and quantitative analysis of systems." Petri Nets and Performance Models, 1993. Proceedings., 5th International Workshop on. IEEE, 1993.

[7] Z1 datasheet, http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf.

[8] IEEE Standard 802.15.4-2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), May 2003.

[9] F. Bouabdallah, N. Bouabdallah and R. Boutaba, "On Balancing Energy Consumption in Wireless Sensor Networks", IEEE Transactions on Vehicular Technology, vol. 58, no. 6, pp. 2909-2924, 2009.

[10] Paulo Rogério Pereira, António Grilo, Francisco Rocha, Mário Serafim Nunes, Augusto Casaca, Claude Chaudet, Peter Almström and Mikael Johansson, “End-To-End Reliability In Wireless Sensor Networks:Survey And Research Challenges” International Conference on Wireless Communications, Networking and Mobile Computing, 2007.WiCom 2007, Page(s).2771 – 2774.

[11] D. Bhattacharyya, T. Kim and S. Pal, "A Comparative Study of Wireless Sensor Networks and Their Routing Protocols", Sensors, vol. 10, no. 12, pp. 10506-10523, 2010.

[12] Wikipedia, “Wireless sensor node”, http://en.wikipedia.org/wiki/Sensor_node, accessed on July 2015.

[13] List of wireless sensor nodes:http://en.wikipedia.org/wiki/List_of_wirelss_sensor_nodes#List_of_gateway_sensor_nodes.

[14] Cao X., Hu F. Wireless Sensor Networks: Principles and Practice, Auerbach Publications Chapter 2

[15] Ayman Z. Faza, Sahra Sedigh-Ali: “A General Purpose Framework for Wireless Sensor Network Applications.” Proceedings of the 30th Annual International

[16] Pressure Sensor: http://www.geokon.com accessed on 02/07/2018.

[17] L. Ho, M. Moh, Z. Walker, T. Hamada, and C.-F. Su, “A prototype on RFID and sensor networks for elder healthcare: progress report,” in Proceedings of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis, 2005, pp. 70–75.

[18] MSP430F2617 datasheet: http://www.ti.com/lit/er/slaz033j/slaz033j.pdf.

[19] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

[20] [Koojana11] Koojana Kuladinithi, Olaf Bergmann, Thomas Potsch Markus Becker, Carmelita Gorg, "Implementation of CoAP and its Application in Transport Logistics," In Proceedings of the Workshop on Extending the Internet to Low power and Lossy Networks (April 2011)

[21] [11]H. Kim, J. Ko, D. Culler and J. Paek, "Challenging the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL): A Survey", IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2502-2525, 2017.

[22] Jiang H., PB-FCFS-A Task Scheduling Algorithm Based on FCFS and Backfilling Strategy for Grid Computing, in Proceedings of the Joint Conferences in Pervasive Computing, Taiwan, China, 03-06 December 2009, pp. 507-510.

[23] Yu Y., Ren S., Chen N., et al., Profit and penalty aware (pp-a-ware) scheduling for tasks with variable task execution time, in Proceedings of the ACMSymposium on Applied Computing (SAC' 10), Michiga, America, 24-28 October, 2010,pp. 334-339

[24] Chen Y. B., Chanet J. P., Hou K. M., RPL Routing Protocol a Case Study: Precision Agriculture, in FirstChina-France Workshop on Future Computing Technology (CF-WoFUCT'12), Harbin, China,16-17 February, 2012, pp. 754-758.

[25] Chen Y. B., Chanet J. P., Hou K. M., RPL Routing Protocol a Case Study: Precision Agriculture, in FirstChina-France Workshop on Future ComputingTechnology (CF-WoFUCT'12), Harbin, China, 16-17 February, 2012, pp. 754-758.

[26] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler. RFC 4944: Transmission of IPv6 Packets over IEEE 802.15.4 Networks. 2007

[27] S. Tilak, N. Abu-Ghazaleh and W. Heinzelman, "A taxonomy of wireless micro-sensor network models", ACM SIGMOBILE Mobile Computing and Communications Review, vol. 6, no. 2, pp. 28-36, 2002.

[28] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler. RFC 4944: Transmission of IPv6 Packets over IEEE 802.15.4 Networks. 2007

[29] IEEE Standard 802.15.4-2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), May 2003.

[30] RFC 7959, The Constrained Application Protocol (CoAP) obtained from https://tools.ietf.org/html/rfc7959#page-6

[31] A. Dunkels, B. Gronvall, T. Voigt, Contiki - a lightweight and flexible operating system for tiny networked sensors, in: Proc. 29th Annual IEEE International Conference on Local Computer Networks, Tampa, Florida, US+A, Nov. 2004, pp. 455–462.

[32] http://www.contiki-os.org/ [Accessed 25 Jul. 2019].

[33] WaqaasMunawar, Muhammad HamadAlizai, OlafLandsiedel, and Klaus Wehrle. Dynamic tinyos:Modular and transparent incremental code-updates for sensor networks. In ICC'1 0, pages 1-6, 2010.

[34] JaeinJeong. Node-level representation and systemsupport for network programming. C S294-1 DeeplyEmbedded Network System Class Project, 2003.

[35] J. JEONG and D. CULLER, "Incremental Network Programming for Wireless Sensors", International Journal of Communications, Network and System Sciences, vol. 02, no. 05, pp. 433-452, 2009.

[36] Mazumder, Biswajit, and Jason O. Hallstrom. "An efficient code update solution for wireless sensor network reprogramming." Proceedings of the Eleventh ACM International Conference on Embedded Software. IEEE Press, 2013.

[37] Panta, R., Khalil, I. and Bagchi, S. (2019). Stream: Low Overhead Wireless Reprogramming for Sensor Networks.

[38] Rajesh Krishna Pant a, SaurabhBagchi, and Samuel P. Midkiff. Zephyr : efficient incremental reprogramming of sensor nodes using function call indirections and difference computation. In Proceedings of the 2009 con ference on USENIX Annual technical con ference, USENIX' 09, pages 32-32, Berkeley , CA, USA, 2009. USENIX Association.

[39] Nirmala, M. B., and A. S. Manjunath. "Mobile agent based secure code update in wireless sensor networks." 2015 International Conference on Information Networking (ICOIN).IEEE, 2015.

[40] Evers, Leon, Paul Havinga, and Jan Kuper. "Dynamic sensor network reprogramming using sensorscheme."2007 IEEE 18th international symposium on personal, indoor and mobile radio communications. IEEE, 2007

[41] Alessandrelli, D.; Patracca, M.; Pagano, P. T-Res: Enabling Reconfigurable in-Network Processing in IoT-Based WSNs. In Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems, Cambridge, MA, USA, 20–23 May 2013; pp. 337–344.

[42] P. Azad and V. Sharma, "Cluster Head Selection in Wireless Sensor Networks under Fuzzy Environment", ISRN Sensor Networks, vol. 2013, pp. 1-8, 2013.

[42] F. Van den Abeele, J. Hoebeke, I. Moerman and P. Demeester, "Integration of Heterogeneous Devices and Communication Models via the Cloud in the Constrained

Internet of Things", International Journal of Distributed Sensor Networks, vol. 2015, pp. 1-16, 2015.

APPENDIX A: Observer Function

The code script shows the modification of observer function (***er-coap-13-oberserving.c***) done on the erbium.

```
void
coap_observe_handler(resource_t *resource, void *request, void *response)
{
coap_packet_t *constcoap_req = (coap_packet_t *) request;
coap_packet_t *constcoap_res = (coap_packet_t *) response;
  uint8_t *host = "aaaaaaaaaaaaaaaaaaaaaaaaa";
  uint8_t *res_path = "rrrrrrrr";
uip_ipaddr_taddr;
uint8_tlen = 0;
  uint8_t res_path_len = 0;
if (coap_req->code==COAP_GET &&coap_res->code<128) /* GET request and response without error code */
{
if (IS_OPTION(coap_req, COAP_OPTION_OBSERVE))
{
if ((len=REST.get_query_variable(request, "host", host)))
{
host[len]=0;
coap_convert_str_to_ip6addr(host, &addr);
if ((res_path_len = REST.get_query_variable(request, "res", &res_path)))
{
res_path[res_path_len] = 0;
}
printf("Agg IP = %s Resource = %s\n Adding Aggregator as Observer. \n ",h, res_path);
if (coap_add_bind_observer(&addr, COAP_DEFAULT_PORT, coap_req->token, coap_req->token_len,resource->url, res_path, res_path_len, 0, 0, &UIP_IP_BUF->srcipaddr, coap_req->observe, NULL))
```

```
{
coap_set_header_observe(coap_res, coap_req->observe);
}
else
{
coap_res->code = SERVICE_UNAVAILABLE_5_03;
coap_set_payload(coap_res, "TooManyObservers", 16);
}}
else
{
if (coap_add_observer(&UIP_IP_BUF->srcipaddr, COAP_DEFAULT_PORT, coap_req->token,
coap_req->token_len, resource->url, coap_req->observe))
{
coap_set_header_observe(coap_res, coap_req->observe);
}
else
{
coap_res->code = SERVICE_UNAVAILABLE_5_03;
coap_set_payload(coap_res, "TooManyObservers", 16);
}}}
else /* if (observe) */
{
/* Remove client if it is currently observing. */
coap_remove_observer_by_url(&UIP_IP_BUF->srcipaddr, UIP_UDP_BUF->srcport, resource-
>url);
} /* if (observe) */
} /* if (GET) */
} * End of function *
```

APPENDIX B: In-Network aggregation

The erbium code script modification in the file ***er-coap-dyn-module.c***

```
void
dyn_module_resource_handler (void *request, void *response, uint8_t *buffer, uint16_t
preferred_size, int32_t *offset)
{
coap_packet_t *coap_req = (coap_packet_t *) request;
coap_packet_t *coap_res = (coap_packet_t *) response;
const char *uri_path;
uint8_t uri_path_len;
uint8_t res_type = 0; /* 1= None, 2 = Input, 3 = Output, 4 = Control, 0 - Error */
uint8_t res_num = 0;
uint8_t dyn_module_name[4];
static uint16_t obs_counter = 0;
uri_path_len = coap_get_header_uri_path(coap_req, &uri_path);
get_dyn_module_info_from_uri(uri_path, uri_path_len, dyn_module_name, &res_type,
&res_num);
if(strncmp(dyn_module_name, dyn_module.dyn_module_name, 3) != 0)
{
printf ("Invalid URI\n");
return;
}
switch (res_type) {
case ALL: /* Uri is /dyn_module_NAME - respond to GET only */
if (REST.get_method_type(coap_req) == METHOD_GET)
{
coap_res->code = CONTENT_2_05;
//respond with values of input and output and control
}
break;
```

```
case INPUT: // Input
if(REST.get_method_type(coap_req) == METHOD_GET)
{
char content[5];
if(res_type == INPUT)
{
coap_set_payload(coap_res,content,snprintf(content,sizeof(content),"%u",dyn_module.in[res_nu
m]));
}
else
{
//coap_set_payload(coap_res,content,snprintf(content,sizeof(content),"%u",dyn_module.con[res
_num]));
}}
if(REST.get_method_type(coap_req) == METHOD_PUT)
{
uint8_ttmp[MAX _PAYLOAD_LEN];
const char url[MAX _PATH_LEN];
size_tlen = coap_get_payload(request, tmp);
dyn_module.in[res_num] = atoi(tmp);
printf("Arrived Input [%u]\n", dyn_module.arrived_inputs);
if(dyn_module.arrived_inputs ==((1 <<dyn_module.num_in) - 1) || dyn_module.separate_inputs)
{
uip_ipaddr_t *addr = &UIP_IP_BUF->srcipaddr;
if(dyn_module.processing_logic(res_num))
{
resource_t *resource = get_resource_by_url(url, strlen(url));
if(resource)
{
if(dyn_module.separate_inputs)
{
dyn_module_resource_notifier(resource, &dyn_module, res_num);
```

```
}
else
{
dyn_module_resource_notifier(resource, &dyn_module, 0);
dyn_module.arrived_inputs = 0;
}
} /*resource*/
} /* if processing */
} /*if arrived_inputs */
} /* if PUT */
break;
case OUTPUT: //output
if (REST.get_method_type(coap_req) == METHOD_GET)
{
uint8_tmsg[MAX_ PAYLOAD_LEN];
REST.set_response_payload(response, msg, strlen(msg));
}
break;
default:
{
printf("Invalid Request\n");
}
} /*Switch */
} /*dyn_module_resource_handler function */
```

APPENDIX C: Remote Deployment

The ***er-dynamic-loader.c*** code script modification done on erbium

```
void
dloader_handler(void* request, void* response, uint8_t *buffer, uint16_t preferred_size, int32_t
*offset)
{
coap_packet_t *constcoap_req = (coap_packet_t *) request;
uint8_t method = REST.get_method_type(request);

if (method == METHOD_PUT)
{
char *filename = "avg.ce";
if (!IS_OPTION(coap_req, COAP_OPTION_BLOCK1))
{
restlet_load_file(filename);
}
else
{
intfd;
uint8_tlen;
uint8_t *incoming = NULL;

fd = cfs_open(filename, CFS_WRITE | CFS_APPEND);
if(fd == -1) {
printf("Unable to open file\n");
const char *error_msg = "Unable to open file";
REST.set_response_payload(response, error_msg, strlen(error_msg));

return;
}
```

```
if ((len = REST.get_request_payload(request, (const uint8_t **) &incoming)))
{
static uint8_t more = 1;
cfs_write(fd, incoming, len);
cfs_close(fd);
printf("Block[%u] = Len <%u/%u>\n", coap_req->block1_num, len, coap_req->block1_size);
if(!(coap_req->block1_more) && more)
{
more = 0;
printf("Done Transferring. Start Loading\n");
restlet_load_file(filename);
}
}
REST.set_response_status(response, REST.status.CHANGED);
coap_set_header_block1(response, coap_req->block1_num, 0, coap_req->block1_size);
}
}
REST.set_header_content_type(response, REST.type.TEXT_PLAIN);

}
```

APPENDIX D: Data Source

The erbium data source function code done on ***er-sensor.c***.

```
PERIODIC_RESOURCE(temp, METHOD_GET, "gpio/btn", "title=\"Temp Sensor\";obs", 5 *
CLOCK_SECOND);

#if !(RES_TEMP_USE_SHT11_DATA)
const static uint8_t data[100] = {
21, 23, 27, 27, 29, 21, 23, 27, 22, 25, 24, 26, 23, 25, 29, 29, 26, 27, 21, 26, 20, 22, 21, 28, 21, 24,
21, 21, 29, 22, 25, 28, 26, 22, 26, 25, 24, 29, 22, 27, 25, 27, 24, 28, 22, 23, 28, 29, 20, 21, 25, 20,
21, 26, 28, 23, 20, 20, 24, 20, 22, 24, 29, 28, 22, 25, 23, 27, 25, 26, 25, 20, 24, 29, 29, 27, 22, 27,
26, 23, 26, 21, 24, 28, 28, 23, 22, 20, 23, 26, 29, 25, 26, 28, 24, 29, 23, 22, 26, 23
};
unsigneddata_iterator = 0;
#endif

void
temp_handler(void* request, void* response, uint8_t *buffer, uint16_t preferred_size, int32_t
*offset)
{
REST.set_header_content_type(response, REST.type.TEXT_PLAIN);
coap_set_header_max_age(response, MAX_AGE);
static uint32_t tmp; // state of temperature resource at the moment
static char msg[11];

#ifdef RES_TEMP_USE_SHT11_DATA // use sht11 sensor for temperature resource
i2c_disable(); //http://comments.gmane.org/gmane.os.contiki.devel/14819
sht11_init();
tmp = (unsigned) (-39.60 + 0.01 * sht11_temp());
#else // use stored values for temperature resource
tmp = data[data_iterator++ % 100];
#endif
```

```
snprintf(msg, sizeof(msg), "%lu", tmp);
REST.set_response_payload(response, msg, strlen(msg));

/* A post_handler that handles subscriptions will be called for periodic resources by the REST
framework. */
}
/*
```

Declaration

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any higher education institution. To the best of my knowledge and beliefs, the thesis contains no material previously published or written by another person except where due reference is made.

Declared by: Zerihun Befekadu

Signature ……. Date ……

Confirmed by Principal advisor: Girum Ketema (PhD)

Signature ……. Date ……

Confirmed by Co-advisor: Worku Birhanie

Signature ……. Date ……