

**JIMMA UNIVERSITY**  
**Jimma INSTITUTE OF TECHNOLOGY**  
**FACULTY OF COMPUTING**

Optimization Of Multiple Conditional Observation  
Relationship On A Single Resource

By: Frehiwot Tekalgne

Advisor: Girum Ketema (PhD)

A THESIS SUBMITTED TO:

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

**JIMMA UNIVERSITY**  
**JIMMA INSTITUTE OF TECHNOLOGY FACULTY OF**  
**COMPUTING**

This is to certify that the thesis prepared by Frehiwot Tekalgne titled Optimization of multiple conditional observation relationship on a single resource and submitted in partial fulfillment of the requirements for the degree of Master Of Science In Computer Networking complies with the regulations of the university and meets the accepted standards concerning the originality and quality.

Frehiwot Tekalgne  
Student



**14/11/2021**

Dr. Girum Ketema (PhD)  
Advisor



**29/12/2021**

Duresa Derasa (M.Sc.)  
Co-Advisor

\_\_\_\_\_

\_\_\_\_\_

Submitted By:

Frehiwot Tekalgne



30/12/2021

**Approved By:** We the examiners board approve that this thesis has passed through the defense and review process.

1. Dr. Henock Mulugeta



January 10, 2022

Examiner 1

Signature

Date

2. \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

Examiner 2

Signature

Date

3. \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

Chairperson

Signature

Date

## Acknowledgment

First and foremost, I would like to Thank and acknowledge God, the almighty, who has granted countless blessing and opportunity all the way, so that I can do my finest to be able undertake my journey.

I would like to express my special thanks of gratitude to my advisor Dr. Girum Ketema who gave me this wonderful opportunity to work on this topic, which also helped me reach to the point where i came to know about so many with tremendous support and help. Am so thank full for the motivation and encouragement – your guidance and inspiration during the thesis process was invaluable, without my advisor encouragement and guidance this thesis would not have materialized.

I also thank the entire jimma university; Computer Networking Department staffs those who did everything with patience to support us in all the way.

Finally, Words are not enough to express my gratitude for my family. Thank you very much for all continuous and unparalleled love, help and support. This journey would not have been possible if not for my selfless family and **I dedicate this milestone to them.**

# *Abstract*

*The Constrained Application Protocol (CoAP) is a simple and lightweight machine-to-machine (M2M) protocol for constrained devices for procedure in lossy networks which offer a small storage capacity and restricted processing benefiting the function of reliable delivery. Its implemented for message exchange over UDP to support IoT providing observing resources. However, CoAP lacks efficiency to perform multiple resources and can't handle frequent requests. This creates the highest issue on energy, packets, storage and more till observers started being observed conditionally which benefits the CoAP observers to avoid unnecessary communication by letting CoAP client to specify the conditions the client interested in but trigger potential CoAP observers registered on observers list to raise optimization issues whenever new additional observer allowed to join makes the request multiple which makes the packet generation leading to observers request and their information requires extensive memory and energy.*

*Since huge amount of data have enough capacity to disturb the entailer network resulting buffer overflow and packet loss. To overcome these problems , we proposed an approach to optimize multiple CoAP clients request in to a single link format by aggregating multiple observation relationships designing user application module and deploying remotely to the sensor nodes on the network to ease high storage need by dropping excessive packet overflow. this implementation carried out using Contiki with Cooja Simulator and evaluated by captivating diverse measures of storage space, energy consumption, delay and number of packet transmitted in the network.*

*The proposed approach improved overall performance of the system by identifying the impact of multiple conditional observation relationship and brought an efficient solution to further reduce traffic.*

**Keywords:** CoAP, IoT, UDP, Contiki OS, Cooja, Conditional Observation

# Contents

CHAPTER 1: INTRODUCTION .....	1
1. INTRODUCTION .....	1
1.1. REGULAR CoAP REQUEST WITH OBSERVE.....	3
1.2. RESOURCE REGISTRATION AND NOTIFICATION METHOD .....	4
1.3. COAP CONDITIONAL OBSERVE OPTION.....	5
1.4. BACKGROUND AND MOTIVATION.....	7
1.4.1. CoAP observe .....	7
1.4.2. CoAP observe .....	8
1.5. MOTIVATION .....	8
1.6. PROBLEM STATEMENT.....	9
1.7. GENERAL AND SPECIFIC OBJECTIVE.....	10
1.7.1. GENERAL OBJECTIVE .....	10
1.7.2. SPECIFIC OBJECTIVE.....	11
.1.8 EXPECTED OUTPUT OF THE PAPER WORK .....	11
.1.9 SCOPE.....	12
.1.10 LIMITATION.....	12
.1.11 METHODS .....	13
.1.12 Literature review .....	13
.1.13 Software and simulation tools .....	13
1.14 THESIS OUTLINE.....	15
CHAPTER 2: LITERATURE REVIEW .....	16
2.1 CONSTRAINED NETWORKS .....	16
2.2 CONSTRAINED APPLICATION PROTOCOL (COAP).....	16
2.3 CONDITIONAL OBSERVE.....	18
2.4 MULTIPLE CONDITIONAL OBSERVATION .....	20
2.5 PACKET AGGREGATION.....	21
CHAPTER 3: RELATED WORK .....	25
3.1 CONSTRAINED APPLICATION PROTOCOL (COAP).....	25
3.2 CONDITIONAL OBSERVATION IN COAP .....	26

3.3	PACKET LOSS .....	28
3.4	EFFICIENT AGGREGATION SCHEME TO MAXIMIZE WSN LIFESPAN .....	29
3.5	ENERGY-EFFICIENT DATA AGGREGATION .....	29
3.6	COMPRESSION AND AGGREGATION.....	29
CHAPTER 4: PROPOSED APPROACH .....		31
4.1	CONTRIBUTION.....	31
4.1.1	Data Aggregation: .....	32
4.1.2	Energy Consumption: .....	32
4.1.3	Traffic Load reduction:.....	32
4.1.4	Save Memory: .....	32
4.1.5	Costly Transmission .....	33
4.2	PROPOSED APPROACH .....	33
4.2.1	CONSTRAINED ENVIRONMENT.....	33
4.2.2	PROPOSED FRAMEWORK .....	34
4.3	DETAILED APPROACH .....	35
4.3.1	REGISTER AND NOTIFY CONDITIONAL OBSERVERS .....	36
4.4	ADD CONDITIONAL OBSERVERS.....	37
4.5	REGISTER AND NOTIFY CONDITIONAL OBSERVERS.....	39
4.6	AGGREGATOR FUNCTION.....	41
4.7	AGGREGATION FUNCTION MATHEMATICAL MODEL.....	45
4.8	PROPOSED APPROACHES PHASES.....	45
CHAPTER 5: IMPLEMENTATION AND RESULTS .....		47
5.1	Experiment Setup.....	47
5.1.1	Contiki OS .....	48
5.2	Contiki Communication .....	49
5.2.1	Sky Mote.....	49
5.2.2	Cooja Network Simulator .....	50
5.2.3	Erbium .....	51
5.2.4	Preliminaries .....	51
DETAILS.....		51
5.3	SIMULATION SETUP .....	52
5.3.1	Simulation Setup Structure .....	53

5.4	Evaluation.....	54
5.4.1	Functional evaluation .....	54
5.5	Mathematical Evaluation.....	59
5.5.1	Packets Records .....	60
5.5.2	Power Consumption .....	60
5.5.3	Delivery Time .....	61
5.5.4	Performance Evaluation.....	62
CHAPTER 6: CONCLUSIONS AND FUTURE WORK.....		67
6	Future work and conclusion .....	67
6.1	Conclusion.....	67
6.2	Future work.....	67
6.3	Future topics .....	68
References.....		69
APENDIX A:.....		70
APENDIX B:.....		71
APENDIX C:.....		76



# List Of Figures

Figure 1. Abstract Layering Of CoAP .....	15
Figure 2: The Successful And Failure Response Results Of Get Method .....	16
Figure 3: Get Request With Observe .....	18
Figure 4 : Two Condition Options To Define In-Side A Range Option 6/5 And 5/15 .....	19
Figure 5: Two Observe Options With Logic Flag .....	21
Figure 6: Network Model For Dc Aggregated Transmission.....	22
Figure 7: Architecture Of Erbium. ....	25
Figure 8: Conditional Observation Module .....	26
Figure 9: Plot Of Packet Loss For Various Motes .....	28
Figure 10: Regular Network Routing . ....	35
Figure 11: Architecture Of Constrained Environment Communication .....	37
Figure 12: CoAP Observe Conditionally, If Condition Met .....	38
Figure 13 : Registration And Notification.....	39
Figure 14:Proposed Conditional Observation Work Flow .....	42
Figure 15: Architecture Of Aggregation Function .....	44
Figure 16: Process Of Aggregation .....	45
Figure 17: Simulation Setup .....	55
Figure 18: Establish Network .....	56

Figure 19: Aggregator Server .....	57
Figure 20: Routing Table .....	59
Figure 21: Client Nodes Output .....	59
Figure 22: Clients Output: Script Editor .....	60
Figure 23: Radio Messages Enabled In Cooja Simulator .....	61
Figure 24: Average Energy Consumption Against Number Of Nodes .....	64
Figure 25: Number Of Packets Transmitted Alongside Nods .....	65

# List Of Tables

Table 1: CoAP Message Format .....	17
Table 2: Condition Option Number .....	18
Table 3: Packet Loss (In %) As A Function Of Packet Generation .....	27
Table 4: Summary Of Literature Review .....	30
Table 5: Summary Of Literature Review .....	36

## List of Acronyms

**6LoWPAN** IPv6 over Low-Power Wireless Area Networks

**CoAP** Constrained Application Protocol

**CoRE** Constrained RESTful Environments

**Contiki** Operating System

**HTTP** Hyper Text Transfer Protocol

**IoT** internet of things

**M2M** machine to machine communication

**OS** Operating System

**REST** Representational State Transfer

**RF** Radio Frequencies

**RPL** Routing Protocol for Low Power and Lossy Network

**URI** Uniform Resource Identifier

**UDP** User Datagram Protocol

**WSN** Wireless Sensor Network



# CHAPTER 1: INTRODUCTION

## 1. INTRODUCTION

The **internet of things** is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. [31].this platform is capable of providing the smart environment bringing sensor usage to a new level consisting of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments. IoT devices share the sensor data they collect by connecting to an IoT gateway or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another.[6] those Sensors devices play an important role in creating solutions using IoT by detecting external information, replacing it with a signal that humans and machines can distinguish to create the highest possibility of collecting data in most any situations and wide range of IoT sensors used to detect and measure various physical phenomena such as pressure, humidity and temperature.[32]

Where in IoT, M2M communication is not only sensors reporting to a server, but true collaboration between things. It is fundamental that all sensor nodes (mote) are truly a part of the Internet; they therefore need their very own IP address. This makes IPv6 an absolutely central enabling technology for the IoT [14]. When all constraints are applied correctly, REST enables architectural properties of key interest. REST is the simplest web service architectures that implementations relying on web services thought to show greater promise for developing the IoT [33]. When applying REST in the context of the IoT and M2M applications, the nodes and their sensors and actuators become abstract resources identified by URIs. In order to address the overhead and the shortcomings of HTTP, the IETF's CoRE working group has developed the Constrained Application Protocol (CoAP). CoAP includes a subset of the functionality of HTTP that is redesigned to suite resource constrained systems, and it also adds other capabilities to address special needs of M2M applications and the IoT. [1]

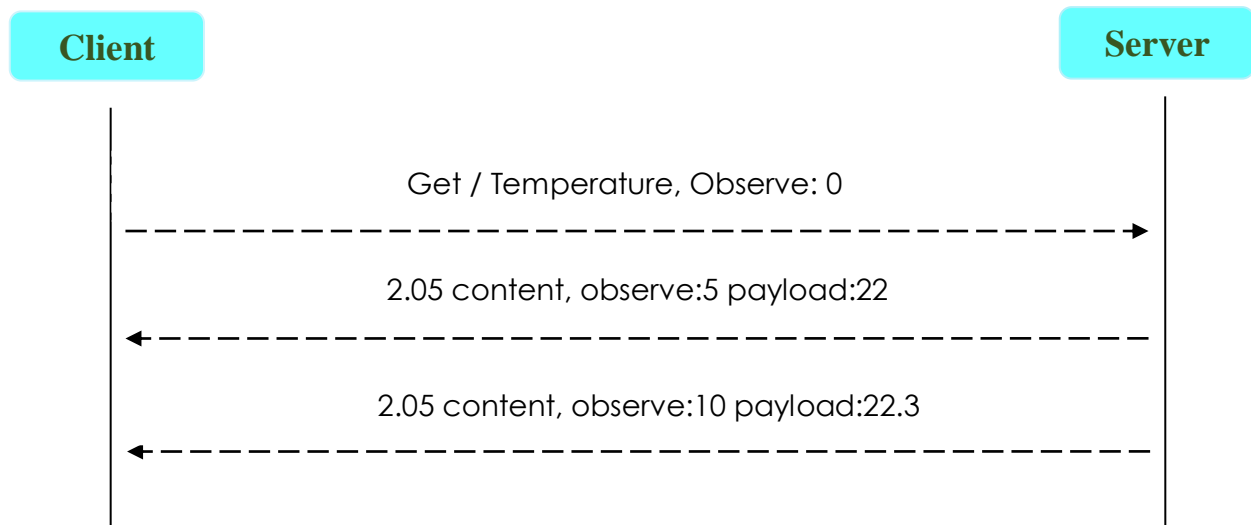
In IoT applications, it is common that clients want to be notified about changes in resources on some or address specified servers. Unfortunately, HTTP has no direct support for, but CoAP has a special addition for this purpose, which provides a mechanism for clients to observe either single or multiple resources on a server. This allows a client to be continuously noticed about changes in the representation of the observed resource in CoAP normal observe. Monitoring and sensing of common physical variables as temperature, pressure or humidity are typical application of WSNs. [2] They are characterized by having slow changes of state and, therefore, require a sensor node to sample their value at low rates. Generally, the information collected by each node is constituted by few bytes and fits the data frames defined by common WSNs standard protocols such as the IEEE 802.15.4 protocol. [10]

CoAP tolerates UDP broadcast and multicast of addressing so that huge sized payloads can be transferred using block-wise transfer where the resource is fragmented in to reduced pieces and sent over multiple CoAP message and through the observe option CoAP clients can observe changes in the state of resources state but the observation can be extended in to conditions to put observers set specific set of condition to observe the resource conditionally. This proposes a chance to outspread network intelligence, improve scalability, and optimize the lifetime and performance in order to satisfy the necessities from the constrained nodes and networks on the process of data exchange (CoAP/Observe).

The use of CoAP Observe for clients to be informed of resource changes, composed with the possibility of incorporating representations, raise optimization subjects when multiple requests for observation exist in a network. These optimization issues with conditional observation involve registration steps, energy saving, consistency and caching. Setting conditions and Registration phases have be wisely put in place, while keeping data consistency and suitability, so that energy could be saved. In Our situation the framework for registration phases composed with the aggregation of notification while observers successfully meets the set of standard conditions for observation, so that energy saving is maximized, bandwidth is used resourcefully and the overall delay is reduced [8].

## 1.1. REGULAR CoAP REQUEST WITH OBSERVE

As the demonstration below describes, in the usual CoAP get request when the client observes a resource on the server then it will receive a response whenever their server updates the resources, as the demonstration every 5 seconds the client will receive a notification response. So after the observe relationship is established and whenever the state of resource changes, the new representation is pushed to the observer. In most cases the server resource changes too often while the client does not want to receive notification soften.



### Observing a resource in CoAP

- **Subject:** is resource positioned at certain CoAP server. state of the resource may change over time, ranging from infrequent updates to continuous state transformations.
- **Observer:** is a CoAP client that is concerned in the current state of the resource
- **Observation Relationship:** client registers with a resource by sending a modified GET requesting server. request bases the server to establish an observation relationship between the client and the resource. The response to the GET request supplies the client with a representation of the current resource state.
- **Notification:** When state of a resource changes, the server notifies each client that has an observation relationship to that resource. The notification is an additional response to the GET request; it supplies the client with a representation of the new resource state.



Clients could be automatically removed from the list of observers when they are no longer interested in the observed resource. In this case, the server determines the client's interest from the acknowledgments of confirmable notifications, so if a client wants to receive notifications after it has been removed from the observers list, it needs to register again.

The Observe Option modifies an initial GET method, requesting the server to add the sender of the Observe petition to the list of observers for a concrete resource. However, if the server is unable to add the client to the list of observers, it will automatically reply with a simple GET request. Furthermore, the server could adjust these options in order to increase the battery lifetime, so in case that the battery level goes below a determined percentage, it is possible for example to reject every Observe request, replying with a simple GET request instead of registering the petition for being notified every time that the resource change its value or state.[22]

## 1.2. RESOURCE REGISTRATION AND NOTIFICATION METHOD

**Resource registration:** client can register its interest resource obtainable by the server by execution a GET operation, including an empty Observe Option. In a successful operation the server should return the response, including an Observe Option as well, notifying in this way that the server has subscribed the client to the local list of observers interested in the value or resource requested, so the client will be, since that moment, notified of every change that takes place over this concrete target.

**Notifications:** further responses from server responding an early GET request. [22]For each notification received, an Observe Option with a concrete sequence number, a Token Option, and a payload of the same media type as the initial response will be included. Notifications can be confirmable or non-confirmable, on this way, if a client could not recognize the token in a notification; it must not acknowledge the message, rejecting it with a RST message. Otherwise, an ACK confirmation message will be send.

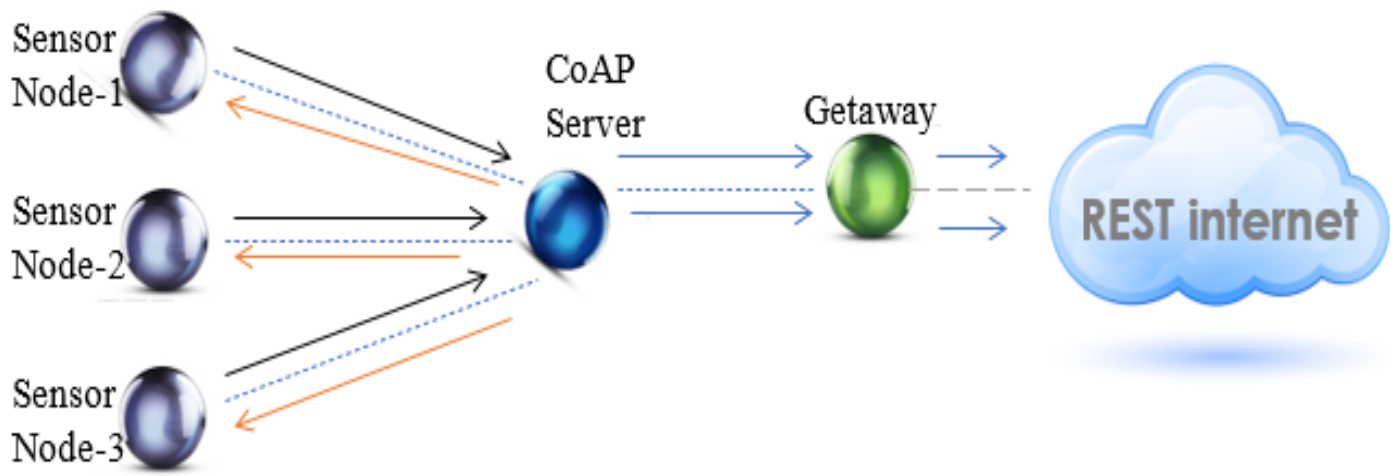


Figure 10: Regular network routing

The above regular constrained network packet routing is put on as an example to demonstrate and to show the difference between the regular and conditional observation network flow. The sensor nodes send information to the nearest node if any or forward to server node and the server sends to gateway the bridge of REST environment to process.

### 1.3.COAP CONDITIONAL OBSERVE OPTION

This observe option lets the clients to observe different resources conditionally by announcing how they can be observed. a mechanism is described to provide additional information to the Observe Option through the use of query parameters. It is possible to define a fixed set of query parameters to enable conditional observations. However, many more query parameters can be offered by a resource for different purposes. This complicates the automatic processing of conditional observations. Also embedding the query parameters in the URI encumbers processing at intermediaries. To alleviate this problem, this draft proposes to use CoAP options to specify timing-related conditions, that is, minimum time interval and maximum time interval, for the notifications. Using options ensures a compact representation and well-defined meaning. For resource value related conditions,

e.g. larger than, smaller than, another mechanism such as query parameters can be used to complement the Minimum-Interval and Maximum-Interval Options.[17]

No.	C	U	N	R	Name	Format	Length	Default
TBD		x	-		Minimum-Interval	int	0-2 B	(none)
TBD		x	-		Maximum-Interval	int	0-2 B	(none)

Table 6: MIN | MAX interval of two condition option

This draft defines two condition options for observe, Minimum-Interval and Maximum-Interval. Both options are elective, and Proxy Unsafe (similar to the Observe Option).

The Minimum-Interval and Maximum-Interval options can only be present in a GET request message and its response message. They must be used together with the Observe Option, since they extend the meaning of the Observe Option. The Minimum-Interval/Maximum-Interval MUST be included in the first notification message if the conditional observation relationship is created successfully. If the server does not support the Minimum-Interval/Maximum-Interval option contained in the observe request, it will ignore the them. The Minimum-Interval Option indicates the minimum time interval between two notification responses sent from the server, even if the resource representation changes.

After sending the previous notification response, the server has to wait for the minimum time interval to expire, before sending the subsequent notification response, if the resource representation changes within the specified minimum time interval. The Maximum-Interval Option indicates that the maximum time interval between two notification responses sent from the server, even if the resource representation does not change. After sending the previous notification response, if the resource representation does not change after the maximum time interval, the server needs to send the same resource representation immediately after timeout of the maximum time interval. This can be used to show the liveness of the server.

## 1.4. BACKGROUND AND MOTIVATION

The Constrained Representational State Transfer (REST) ful Environments (CoRE) working group has designed and developed Constraint Application Protocol (CoAP) which is intended to operate in constrained IP networks and provides RESTful services in constrained devices. The CoAP protocol has been developed as a replacement of HTTP for the IoT. After CoAP introduced, observe have been proposed for RESTful services to be provided. In CoAP normal observe the observers store and reuse notification as long as the observation time expiration is still fresh, whenever other clients requesting same resource without requesting it from server. However, the default normal CoAP observe considered to be unable to effectively perform group communication and observe resources, it can't handle speedy , frequent requests. The drawback of CoAP clients normal observe triggered the impression of condition extension to be introduced in to the resource observation process, the condition option has to be used in combination with the observe option and used in both request and response. Even if the launch of conditional observation avoids transmission of unwanted to CoAP client's notification only when the set condition is met, CoAP server will send a notification response with the latest state change. But in the cases of when multiple temperature, pressure or humidity resources conditionally observed, an increased number of observation which is multiple conditional observation leads to an increase in memory space requirement. This limits the number of CoAP clients or observers allowed to register simultaneously

### 1.4.1. CoAP observe

The specialized web transfer protocol which is constrained application protocol has been designed and developed to be a lightweight HTTP” so that it can be suitable to operate in the constrained IP networks. The CoAP communication model is alike to the client/server model of HTTP: a CoAP client subjects a request message to a server and if the CoAP server is able to assist the request, it responds to the requester with a response code and the payload. [24] Contrasting to HTTP, CoAP requests and responses are exchanged asynchronously on top of an unreliable datagram-oriented transport protocol (e.g., UDP). The CoAP messaging model supports 4 types of messages: CON (confirmable), NON (non-confirmable), ACK (Acknowledgement), RST (Reset). Every CoAP message carries a Token whose value is a sequence of 0 to 8 bytes. The Token correlates a response with a request, along with the additional address information of the corresponding CoAP endpoint.

The CoAP client generates a Token for a request message and the server uses the same Token in the response. Each message also contains a 16-bit message ID, which is used to detect message duplicates.

#### 1.4.2. CoAP observe

Constrained application protocols operate under a similar transaction model, CoAP observer may send single or multiple requests for the observable, which provides a response for CoAP request. for the reason the request and response transaction is not happen to be by previously established connection but will be traded asynchronously over CoAP message. When multiple observation requests exist then IF-match option may be used to brand a request conditional on the existing or value of ETag for one or more representations of target resource.

### 1.5.MOTIVATION

CoAP is intended by the REST architecture and is elevated for M2M communication in constrained environments which is broadly desired in resource monitoring. On those applications requesting periodically is not optimal considering values may not vary frequently which triggers unwanted packet transmission. Introducing a method that triggers transmission, only if conditions set by observers changes occur so called conditional observation was the uplifting. But the constraint application protocol (CoAP) observation and the condition extension together with the highest possibility of enormous observers elevate optimization issues, when there are multiple notification requests requiring broad memory and computational capabilities. As memory is a very limited resource of constrained devices , then size of packets transmitted and communication will be affected highly and this will limit multiple observers to get worthy access .

Considering serving conditioned multiple observations puts storage and data at risk, we tuck the gap as an inspiration to propose an approach to optimize potential resources by aggregating together in to a single CoRE link format composition. Even if our key inspiration is to create optimized multiple conditional CoAP observation transaction in order to provide reliable communication set of best practices used to improve various indispensable performance. In this paper we also motivated to deliver accurate main protocol that is used to implement IoT. But more precisely, a variety of tools and techniques be used to monitor and improve network performance such as load balancing, minimize latency, packet loss monitoring, saving the amount of data stored or transported through constrained network devices memory because of the large amount of transferred data have an ability to disturb the network performance. the disruption of network performance in sensor networks causes the highest

terrible on conserving energy and the extreme use of energy or maximum power consumption accelerates the smallest lifetime of network devices. In addition, reduction of Tx and Rx for the reason having Rx and Tx reduce a high packet drop rate.

## 1.6.PROBLEM STATEMENT

The nature of resource-constrained networks positions new design challenges on a lightweight web transfer protocol designed based on the REST architecture standardized by the Internet Engineering Task Force (IETF) to encounter and accommodate the necessities of the constrained Internet of Things (IoT). in a resource-constrained lossy network with various frequencies of transaction is a significantly challenging task that needs to be addressed. CoAP Observation of request and response mostly triggers optimization problems whenever multiple CoAP observation request issued on the network and the same is true for sending multiple notification which highly tumbledown the memory and energy giving elevated further packet transmission with extra consumption of bandwidth whenever there is no existence of ether packet aggregation or conditional observations. This load of frequent responses for a single observer request in ether normal observe or condition al observation , sensor nodes are mostly positioned and exposed as the transmission deeply affected by being break open the chances of packet loss because of multipath fading which highly affects communication and signals on frequencies in many ways. For the reason that sensors and actuators have low power and memory capacity, then optimization of energy, bandwidth and memory throughout the transmission of in both CoAP server and observers but more specifically if additional observer is authorized to join observers list, in other words if multiple observation trendy, is the highest issue in Constrained Application Protocol resource observation. This optimization problem accelerates issues in CoAP resource and subject registration with server's notification while conditional observers are being registered on observers table as a single observer requiring response only when state changes if only he observation expiration time stays fresh. Considering all that, the most critical problems we like to address are

**Memory Space:** allowing conditions in CoAP observation supports multiple access to the resource this increase memory space requirement, as every CoAP clients information needs to be stored in memory then this makes the subject to receive the maximum a memory needed to store potential observers and this will limit multiple observers to access the resource simultaneously.

**Power Consumption:** conditional observation consenting of multiple resource access leads to an increased power consumption at each node and latency. Therefore, as observers notification goes through the process in to their destination creates extra additional packet for every notification.

**Number of Packet:** as numerous CoAP observers allowed observing, the packet generation will also be increased marvelously .through this process of notification or number of packets being generated observers response gets higher.

**Energy:** the intermediate nodes are needed for the scalability and they act as cluster head pausing on behalf of the observers registered on it, requesting resource notifications, as a client, to subject components at CoAP servers. Depending on the maximum age value at notifications from multiple/different subject components, a single aggregated transmission may be done, containing all data, toward a client or, instead, multiple transmissions toward different destinations can be scheduled for transmission in batch, avoiding multiple wakeups. This increases energy saving.

**Network performance:** since over-all performance depends on the traffic strength in the IoT which is lightest for multicast strategy and highest for POST/GET design. This various umber of sensors trying to communicate once triggers the factors that are affecting the network performance and quality of service causing number of transition errors which all leads to network latency.

**Tx/Rx :** the high load of Tx and Rx increase amount of traffic load that is going form our network so managing this issue help us to reduce high packet drop rate and memory space.

## 1.7. GENERAL AND SPECIFIC OBJECTIVE

### 1.7.1. GENERAL OBJECTIVE

The general objective is to make multiple conditional observation relationships in to a single resource by allowing additional (new) observer to send observation request and aggregating multiple “observers” request in to distinct “subject” to optimize the request and notification while CoAP observers establish relationship.

### 1.7.2. SPECIFIC OBJECTIVE

The specific objectives of the involvements that need to be deal with in portion:

- To design user application module
- Design an efficient solution to aggregate multiple relationships into a single to further reduce traffic.
- Modify observe function to allow adding new user to request observation.
- Optimize numerous observation messages in to single by aggregating relationships.
- Develop approach to save time and costly transmission on request/notification
- Reduce CoAP client's memory usage while being registered on table as an observer
- Develop a technique that Check observers to make sure if the upcoming observer (new) if observation from source to same resource existed.
- Deploy a mechanism if multiple CoAP observer request happens to be on table satisfying terms for condition then to be aggregated by adding additional parameter to existing relationship.
- Configure methods for those observers way far from satisfying terms of condition by adding new relationship.
- Registration, allow additional new observer to observe conditionally
- Notify observers, letting the conditional observers on list of observers be notified the state changes

### 1.8 EXPECTED OUTPUT OF THE PAPER WORK

The expected output of our work is improving overall performance of the system by optimizing conditioned resources by aggregating multiple resources together into single CoRE link format composition. The prior achievement of our work is data aggregation for the reason that while we succeed aggregating resources then the network lifetime grow by decreasing energy intake and the optimization will be full after we reduce memory space by limiting unwanted packet overflow.

On this thesis work we finally expect our conditional observation to have the following output:

- Better network life time
- Low energy intake
- Reduced memory space
- Limited packet transmission
- Allowing new additional observer for simultaneous observers registration



- Single Conditioned resource notification out of multiple CoAP clients observation request

### .1.9 SCOPE

This paper work mainly focus on scheming an efficient solution to aggregate multiple observation relationships in to a single one by design and modifying user application module then deploying the application module remotely to the potential sensor nodes on the network to reduce memory space by ether reducing or limiting unnecessary packet transmission and benefiting the energy to maintain the intake for long lasting network life time. All this makes the optimization of multiple CoAP client request much maintainable in addition to the simulators and programing platforms to study the impact of multiple conditional observation relationships on a single resource.

### .1.10 LIMITATION

It is obviously known that optimization of multiple conditional observation requests and merging in to a single rate by deploying aggregation is not the only solution design to overcome the problem of traffic load and crowd but there is also a solution so that aggregation of relationship of sequencing and scheduling in order to organize and maintain the traffic. But the smarter and the last big initiative that I have in mind to work with in the future is in the outside of constrained device or environment. Specifically, working all the optimization process from the outside of the constrained device or environment by deploying or installing getaway and proxy to each other by the technical way of virtualization to easing traffic load while communication of client and sensor servers like that of multiple request and aggregated single response.

More precisely this **limitation** on the research work

- we used network simulator that allows us to run , test and evaluate our uses on fully emulated hardware devices to be Extensive support in the evaluation and optimization providing flexible platform assisting diverse network elements, protocols, traffic, and routing types, whereas under multiple reasons including the cost and the impossibility of getting the real test bed was one of our key factor.
- Implementing aggregation to achieve efficient data dissemination.
- Having hard time on choosing the proper protocol
- And also hardware and intermediate to integrate

- Finding proper simulation and system software's and requirements It does not deal with single conditional request and
- Does not deal on security issue

#### .1.11 METHODS

To meet the goal of the generalized and specified objectives the specific procedures or techniques used to process the main aim in this research paper to critically evaluate our study's overall validity and reliability, the subjects explained as follow:

##### .1.12 Literature review

The method we used is survey of journals articles, scholarly books and other sources of research of theoretical concept for research of specific topic

##### .1.13 Software and simulation tools

◆ **Contiki-OS** is such a platform which is developed to afford this liberty to execute dynamic loading and unloading of codes smoothly. Contiki is established with an event-driven kernel which accommodates selective preventive multi-threading for each exclusive process. It is established on C language which supports multiple environments to configure several micro controller architectures containing MSP430, Atmel AVR, now based on ESB platform. The ESB adopts MSP430 with 2kb RAM and 60 kb ROM, operating at 1 MHz The MSP430 is capable of performing optional reprogramming of on-chip flash storage. Contiki features Some of the key features of Contiki are illustrated below [12].

- Memory allocations an efficient way to provide a mechanism for memory allocation.
- Full IP networking affords a full IP network stack, with standard IP protocols such as UDP, TCP and HTTP.
- Dynamic module loading dynamic loading and linking of modules at run-time is supported.
- Power awareness operates in extremely low power systems which are designed to run for years on a pair of AA batteries.
- The Cooja network simulator makes simulation tremendously easier by providing a simulation environment.

- The Rime Stack supports simple operations such as sending a message to all neighbors or to a specified neighbor, as well as more complex mechanisms such as network flooding and address free multi-hop semi-reliable scalable data collection.
- The Contiki shell provides an optional command line shell with a set of commands that are useful during development and debugging of Contiki systems.
- Proto threads is a mixture of the event driven and the multi-threaded programming mechanisms. With proto threads, event handlers can be made to block, waiting for events to occur.

A running Contiki system consists of the kernel, libraries, the program loader, and a set of processes.

In this section, we clarified the foundation of packet aggregation in order to improve energy efficiency and resource conservation. Additionally, overview of some previously proposed methods and techniques to implement packet aggregation. Finally, we explained used tools including like that of Contiki and how these are related with this thesis work in order to achieve the desired outcomes according to the requirement specifications.

- ◆ ***Constrained Application Protocol (CoAP)*** is a specialized Internet Application Protocol for constrained devices, as defined in RFC 7252. It enables those constrained devices called "nodes" to communicate with the wider Internet using similar protocols. CoAP is designed for use between devices on the same constrained network (e.g., low-power, lossy networks), between devices and general nodes on the Internet, and between devices on different constrained networks both joined by an internet. CoAP is also being used via other mechanisms, such as SMS on mobile communication networks.

CoAP is a service layer protocol that is intended for use in resource-constrained internet devices, such as wireless sensor network nodes. CoAP is designed to easily translate to HTTP for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and simplicity. Multicast, low overhead, and simplicity are extremely important for Internet of Things (IoT) and Machine-to-Machine (M2M) devices, which tend to be deeply embedded and have much less memory and power supply than traditional internet devices have. Therefore, efficiency is very important. CoAP can run on most devices that support UDP or a UDP analogue

- ◆ **COOJA Simulator** *Cooja* Simulator is a network simulator specifically designed for Wireless Sensor Networks. A summary of how Cooja executes binaries of different platforms. Helps routing and compression protocol works in real time simulation to validate the goal set to be meet have succeeded.

#### 1.14 THESIS OUTLINE

Chapter 1: this chapter introduces describe the large problem to be solved and the problem areas which are about to address. Chapter 2: related work discussing subject specific after general introduction using the previous studies to explain to reinforce the study .Chapter 3: literature reviews of the existing research in this problem area answering the question of how the previous researchers dealt with related problem

Chapter 4: proposed work have defined our intended contribution to solving this problem and make the promise of the current work on solving the problem Chapter 5: in implementation and results we tried to present our findings with outcomes Chapter 6: in this chapter we summarized our finding by answering the research question having our major contribution and discussing direction for future research.

## CHAPTER 2: LITERATURE REVIEW

On this section we use the previous literature to generally introduce and discuss subject specific literature in addition to identifying knowledge gaps based on previous literature containing comprehensive contents mostly related suggested techniques, literatures and outlines by numerous researchers. Accordingly, methods and approaches of technologies will be briefly described.

### 2.1 CONSTRAINED NETWORKS

These networks are composed of a significant portion of constrained nodes. Mostly, these constrained node networks are deployed in the edge network of an IoT system. Constrained node networks are deployed in the edge network of an IoT system. As it is a web transfer protocol, it is based on REST full architecture which provides a request/response interaction model between application endpoints and supports built-in discovery of services and resources. Like HTTP, Servers make resources under URL and clients access those resources using methods such as GET, PUT, POST and DELETE of the CoAP protocol.

### 2.2 CONSTRAINED APPLICATION PROTOCOL (COAP)

“The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks in Internet of Things. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation. CoAP is however a single protocol, with messaging and request/response as just features of the CoAP header.

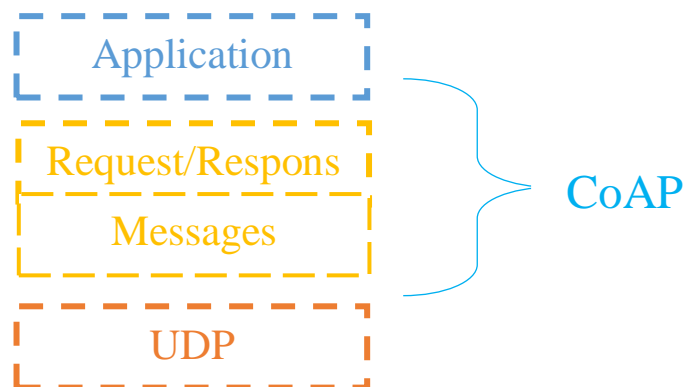


Figure 1. Abstract Layering Of CoAP

CoAP follows a client-server communication model. Client makes request to the server and the server sends back the responses to the client. Client can GET, PUT, POST or DELETE the resources on network. CoAP improves the HTTP request model with the ability to observe a resource. In HTTP, the server needs to do polling again and again to check where there is any state changes to the client or not. Whereas in CoAP, the observe flag is set on the CoAP GET request, the server continues to reply after the initial document has been transferred.

For Quality of Service (QoS), Requests and response messages may be marked as confirmable or non-confirmable. Confirmable messages must be acknowledged by the receiver. Non-confirmable messages are “fire and forget” type.[24]

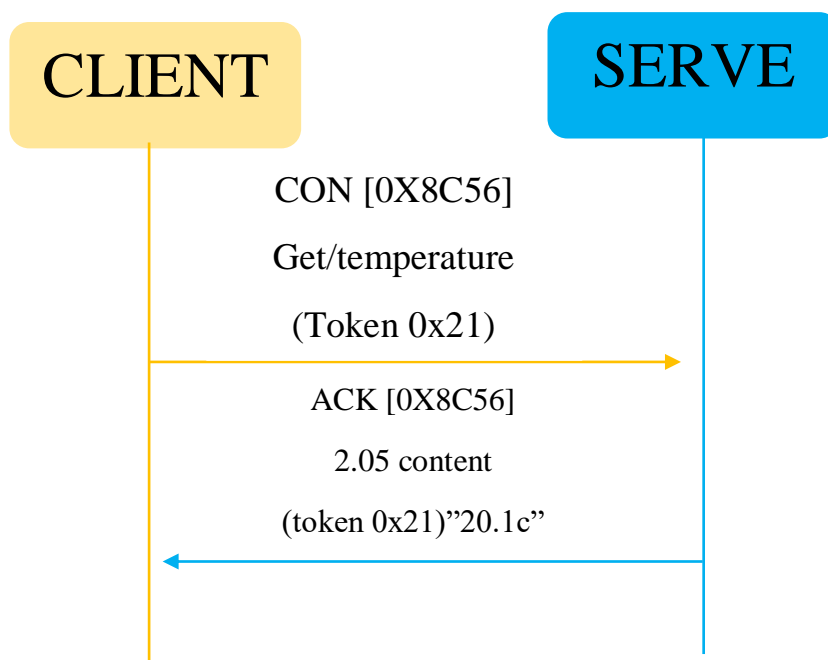


Figure 2: The successful and failure response results of GET method

Piggy-backed: Client sends request using CON type or NON type message and receives response ACK with confirmable message immediately. In fig. 2, for successful response, ACK contain response message (identify by using token), for failure response, ACK contain failure response code. [24]

## Message Format

CoAP is based on the exchange of compact messages that, by default, are transmitted over UDP (i.e. each CoAP message occupies the data section of one UDP datagram) Message of CoAP uses simple binary format. Message= fixed-size 4-byte header plus a variable-length Token plus a sequence of CoAP options plus payload. The format is shown in Table 1.

0		1				2				3											
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Ver		TKL				C				CODE				MESSAGE ID							
oken (if any TKL byets)																					
ptions (if any)																					
ayload (if any)																					

Table 1: CoAP Message Format

With the representation of

- 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)

### 2.3 CONDITIONAL OBSERVE

To avoid transmission of unwanted notifications to clients, the authors of this paper have proposed a new CoAP option ‘Condition’ as an extension to the Observe Option in order to support conditional observations . This option can be used by a CoAP client to specify the conditions the client is interested in. Now, only when the condition is met, the CoAP server will send a notification response with the latest state change. When the condition is not met, the CoAP server will not send the notification response. The following figure demonstrates operation of conditional observation.[14]

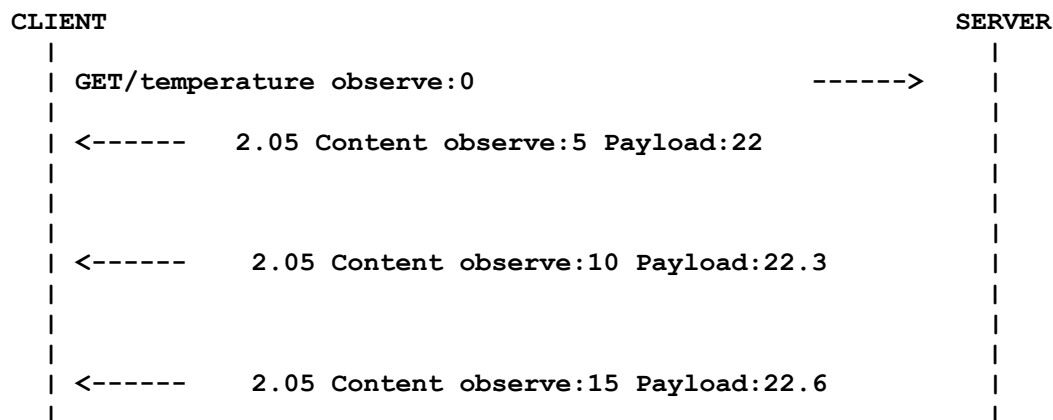


Figure 3: GET request with observe

When a server receives such a request, it first serves the request like a GET request without this option and, if the resulting response indicates success, establishes an observation relationship between the client and the target resource. The client is notified of resource state changes by additional responses sent in reply to the GET request to the client.

CoAP is used for Constrained Networks, especially used for transporting sensor data. Different sensor equipment's have different properties. As such, when a client wants to collect information from a sensor, it does not want to receive too many notification messages within a short time period, if the state of a server resource changes too often. Also, if the resource's representation does not change for a long time, the client wants to receive notifications in order to make sure that the observe relationship is still alive.[15]

### The Condition Option

Type	C/E	Name	Data type	Length	Default
26	E	Condition	uint	1-5 B	(none)

Table 2: Condition Option number

In the response to the initial GET request message, the Condition Option, together with the Observe Option, indicates that the client has been added to the list of observers and that notifications will be sent only when the resource state meets the condition specified in the Condition Option. In all further notifications, the Condition Option identifies the condition to which the notification applies. [15]





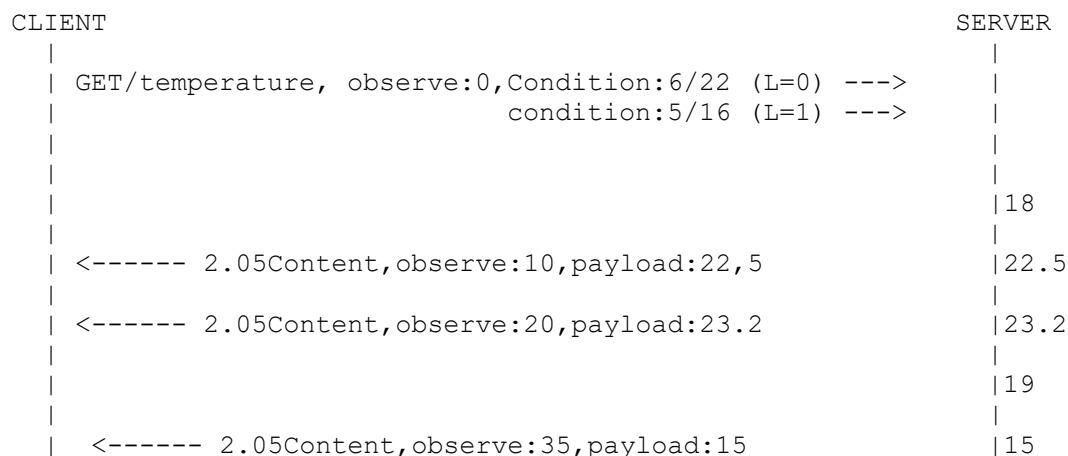


Figure 5: Two Observe options with Logic flag to define out-side a range 6/22 OR 5/15.

## 2.5 PACKET AGGREGATION

In WSN, nodes are energy-constrained and resource-limited and in the process of monitoring and gathering data, each node has a data packet which is supposed to be sent to the sink. Packet aggregation in this context is used to reduce the protocol overhead by appending multiple packets together and send them in a single transmission towards the sink which will reduce the number of transmissions and overhead, associated with each transmission unit.

Aggregation schemes depends on the topology of every network. For instance, in cluster based networks, aggregation is usually performed at CHs while inside a cluster, relay nodes behave as aggregators [13]. In some fundamental features are presented which include latency and energy consumption. At first, latency is actually a duration required to finish the aggregation between aggregator and the source node. It includes distance between nodes, packet size and channel state.

Accordingly, in working mode of data aggregation, aggregator waits for all the data coming from various sources to execute the aggregation. Accordingly, the cut off for residual energy is examined regularly to elect the new aggregator with maximum possible energy within one cluster. [11]

Heretofore, a lot of research has been done in the context of packet aggregation in WSNs by focusing on different aspects of aggregation, for instance, data gathering algorithms using energy metrics in [36], energy aware data aggregation in [12], energy efficient recoverable concealed data aggregation in [13], etc. In this section, we are going to briefly analyze some of the technique and

schemes about aggregation, proposed by academia. Additionally, there are some techniques which not only discuss aggregation but add more value by providing some favorable algorithms like packet-compression, encryption, compressed sensing. This scheme saves energy by bypassing the redundant data while sending concurrently. Data transmission throughout this scheme is made confidential by applying mykletun et al.'s encryption. Therefore, redundancy is removed which results a noticeable decline in consumed energy[6].

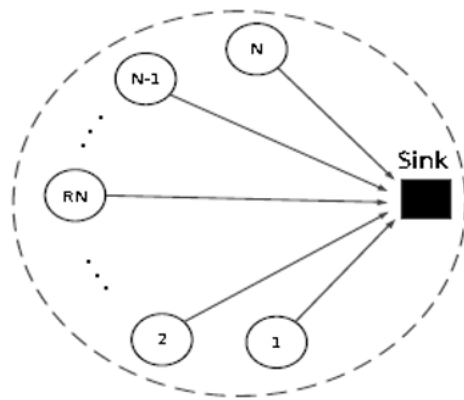


Figure 6: Network model for DC aggregated Transmission

## Review on Literature Review

No	Related work	Approach	Proposition	Reference /Authors
1	Conditional Observe	The condition option can be used by a CoAP client to specify the conditions the client is interested in. Now, only when the condition is met, the CoAP server will send a notification response with the latest state change. When the condition is not met, the CoAP server will not send the notification response.	Condition Option, together with the Observe Option, indicates that the client has been added to the list of observers and that notifications will be sent only when the resource state meets the condition specified in the Condition Option. In all further notifications, the Condition Option identifies the condition to which the notification applies	Li ST, Hoebeke J, Jara AJ: Conditional Observe in CoAP: draft-li-core-conditional-observe-03, IETF Trust. ; 2012
2	Multiple Conditional Observation	Multiple Observe options that are found to be on range been simplified from two messages to only one message with two options, and type field of minimum response time with step and range defining the condition type whenever the client has multiple conditions for notification requirements	Allowing CoAP resource multiple observe request of multiple CoAP observation suggesting alternative approach to multiple conditional observation.	Castro, M., Jara, A., and A. Skarmeta, "Architecture for Improving Terrestrial Logistics Based on the Web of Things", Sensors 12, no. 5, 6538-6575, 2012, May 2012.

3	CoAP Observation	In CoAP Observe , the observe flag is set on the CoAP GET request, the server continues to reply after the initial document has been transferred. This allows servers to stream the state changes to clients as they occur. Any end can stop the observation.	Client makes request to the server and the server sends back the responses to the client. Client can GET, PUT, POST or DELETE the resources on network. CoAP improves the HTTP request model with the ability to observe a resource. In HTTP, the server needs to do polling again and again to check where there is any state changes to the client or not.	[Z. Shelby13] Z. Shelby, Sensinode, K. Hartke, "Constrained Application Protocol (CoAP)," draft-ietf-core-coap-18. [2013-06--28] <a href="http://tools.ietf.org/html">http://tools.ietf.org/html</a>
4	Packet Aggregation	In working mode of data aggregation, aggregator waits for all the data coming from various sources to execute the aggregation. Accordingly, the cut off for residual energy is examined regularly to elect the new aggregator with maximum possible energy within one cluster.	there are some techniques which not only discuss aggregation but add more value by providing some favorable algorithms like packet-compression, encryption, compressed sensing. This scheme saves energy by bypassing the redundant data while sending concurrently.	Zechinelli M JL, Bucciol P, Vargas-Solar G. Energy aware data aggregation in wireless sensor networks.

## CHAPTER 3: RELATED WORK

In this section we use previous studies to explain or reinforce our findings consisting of encyclopedic summary of related proposed techniques and schemes by various researchers. We are going to explain some of those techniques to give a brief knowledge to readers about the efforts which are already presented.

### 3.1 CONSTRAINED APPLICATION PROTOCOL (COAP)

The logical model in this paper work introduces terms to evaluate both application layer packet loss and latency in diverse types of traffic empowered by CoAP. In general, the literature we reviewed related CoAP literature is :

The first thing it concern about is CoAP traffic experimental performances with latency in different situation and their network configurations and the second is to examine the belongings of traditional CoAP congestion control and existing development advices when application constraints (including throughput) are measured.

[25] measure the delay and loss in a CoAP based on WSN with the multihop routing. And in the [26] the delay with loss and quantity of a extremely dishonored wireless network are measured and both the CoAP and MQTT performances are compared. With that [27] the authors mainly compare congestion control mechanisms triggered by CoAP and the delay or the loss for different network situation have to be measured. In [28] the writers suggest an substitute structure to assess performance in the framework of high loss WSNs.

All the literature's we reviewed so far which is directly relate with the application layer protocol of Constrained Application Protocol CoAP papers the min focus area of consideration is on the performance and a success of mathematical model that can be used to measure latency with packet loss on a definite network parameters in addition to quality matrices for the tuning and maintaining of CoAP restrictions.

For the reason that maintaining and tuning the delay or the latency of transition within resource and subjects could lead to the highest reduction traffic lead and increased advantage of better performance.

## 3.2 CONDITIONAL OBSERVATION IN COAP

CoAP is a REST full application protocol for constrained nodes and networks. Through the Observe option, clients can observe changes in the state of resources and obtain a current representation of the last resource state. This document defines a new mechanism in CoAP Observe so that a CoAP client can conditionally observe a resource on a CoAP server, only being informed about state changes meeting a specific condition or set of conditions. This offers possibilities to extend network intelligence, enhance scalability, and optimize the lifetime and performance in order to address the requirements from the Constrained Nodes and Networks [3]

In [1] defines standard set of mostly used conditional observation with its well-known advantages .in every way of CoAP observers those defined well can possibly observe various resources conditionally. Though the resource can obviously notify how the clients possibly be observed, facilitating machine processing of its information. In addition to the impression of conditional observe the head of clusters or the intermediate can process multiple conditional observation.

When ever the CoAP client like to issue an observe request for the available request ,if the standard set failed to appear or exist maybe not defined ye then the observe will fail back to normal or then developer can set novel standard to take advantage of the benefits. The developers of [1] stated using the condition option guarantees a compacted representation and well-defined meaning as the draft defining multiple condition options of CoAP observe accommodating the MIN | MAX interval. The uppermost goal of developers of this exertion is to reduce a costly way of packet transition in highly dynamic environment putting in consideration of power consumed when processing, transition and listening arise in all over the network as suggested ,subscribing to a very specific events only like that of conditional observation offering non complex way of choosing this conditional observe functionality by embedding it in the CoAP protocol as a new CoAP option

On the implementation of conditional observation the developers used [1] erbium(Er) - a low-power REST engine for Contiki. The erbium REST engine is in use for this development for the reason that it accommodates a CoAP implementation supporting CoAP drafts 03,07,12 and 13. The CoAP implementation can be extended to support the new condition option and provided some resources that allow conditional observations.

Figure 6 shows the high-level architectural diagram of Erbium running on a server node handling normal or conditional observation. The architecture consists of several components, namely, the resources, the REST engine, the CoAP (and/or HTTP framework), and optional modules such as (Conditional) Observe Module. The REST engine is responsible for initializing the CoAP framework to store a list of activated resources and to communicate with the optional modules. A conditional observe request received by the CoAP framework will be handled by a service callback function which is declared in the REST engine. The REST engine uses the corresponding handler function to access the states of the resources. As the request is an observation request, the client needs to be registered as an observer in the Conditional Observation Module for future notifications by calling a Post Handler function. The generation of the first response and subsequent notifications are handled by the CoAP framework. For subsequent notifications, the registration of a single observer will trigger the activation of a function that periodically checks for resource state changes and informs all registered observers. The period is defined for observable resource separately upon initialization of the resource[1]

After CoAP observer sends request that is set to be conditional then the CoAP framework will receive it. After conforming GET request then it will callback function in the REST engine. In receiving of request the REST engine the following

- First response will prepared by using the handler of predefined function and calls a post handler function to add the observer to the observer list.
- For each observer, the IP address, port number, URI, and refresh timer are stored for normal observe, while for conditional observation also condition information, last notified value and last notification time are stored.
- The REST engine then periodically checks for resource states and calls the Notify Observer function (which is part of the Conditional Observe Module) to check if the new value satisfies the filtering criteria set by the client.

If it does, the CoAP framework sends the notification to the respective observer(s). Similarly, if a client wishes to stop an observation relationship, it sends a normal GET request to the specific resource which will be received by the CoAP framework and will be sent to the Conditional observe module to be removed from the list.



In general, this paper have introduced the observation option of condition to extend observe option. They have also made the comparison between conditional and normal observe by implementing functionality on a constrained device in addition to theoretical evaluation. finally the result of exponential and theoretical evaluation and comparison , having conditional extension is found to be very essential and useful to the basic observe behavior, both from an application point of view and from a network efficiency point of view.

### 3.3 PACKET LOSS

As the current paper pointed out that whenever Get request sent by CoAP observer through transition scenario, which are not acknowledged will help us to determine the packet loss. As [29] Packet losses arise in situations where there is atmospheric noise existing, and this can be simulated in Cooja by varying the Tx/Rx ratio (in %). The CoAP retransmission factor is set to 0 and the statistics are as shown in the table below,

This parameter is found straight from the CoAP client as the quantity of ineffective packet transmissions. In figure 8 below As graph showed the developers illustrates this difference of packet generation rate and packet loss showed through five hops.

The acceptable limits for these parameters is not definite and is a trade-off between energy consumption and performance. This compensation depends upon the application in mind and its performance requirements

In this paper the developers evaluated three different parameter so called throughput, end to end delay and the packet loss for the various motes in the network. This evaluation also tests the bounds of the RPL routing protocol used in 6LoWPAN, as the scenario features motes with 1 to 5 hops to the router. There is a lot of data generated to maintain the DAG in the RPL protocol for which the ICMP is used as a transport. The initial network traffic is offered by the ICMP protocol trying to establish the DAG, after which it reduces gradually. Also, the resilience of the network was tested for dynamic changes in the topology and the RPL successfully adjusted the routes to ensure packet transmissions. We also demonstrated the reliability of the Californium (Cf) framework, which ensured that negligible packet loss occurred even when the packet generation rate was as high as 1000 packets/sec for a distant mote 5 hops away [30].

The packet generation rate was varied from 1 packet per second up to 100 packets per second in a constant delay by the Java client. These GET requests were sent to the temperature and humidity sensor motes running CoAP servers and the corresponding throughput was noted in each of the cases which is depicted in the table demonstrated.

### 3.4 EFFICIENT AGGREGATION SCHEME TO MAXIMIZE WSN LIFESPAN

An adequate energy-aware scheme is presented in [2] where a network is formed by implementing multiple sensors which are systematically gathering information and forwarding this data towards the sink. At the same time, these sensors are behaving as aggregators due to their capability of executing data aggregation within the network. Consequently, this aggregation process decrease the time required to send data from source till destination which is referred in [2] as sensor's life span. In the described technique, called MLDA algorithm, a nearly ideal polynomial time algorithm is presented with some clustered based advance technique to enhance the lifespan. After all the experiments, results showed that MLDA perform 1.15 - 2.32 times better than the current aggregation schemes in limited area networks. Additionally, clustered based approach gained 2.61 times better efficiency in life span than the typical ones.

### 3.5 ENERGY-EFFICIENT DATA AGGREGATION

Many techniques and algorithms have been presented to conserve energy and resources in WSNs. In this section, three techniques are described to decrease the energy consumption in WSNs. In [15], such an approach is given which proposes a framework, acts as a middleware for data aggregation inside a network in SPIN. It also shows a method to send minimal data with less possible error. In this scenario, nodes are forwarding packets towards sink, at first, there is broadcast message about packets from A for all the neighboring nodes and on reception of request from B, packets are transmitted towards the node B which is intended to receive the packets.

### 3.6 COMPRESSION AND AGGREGATION

Compression is another technique which is used to reduce overhead, produced by large amount of data travelling. where data is sent in narrow compressed way rather than transmission of whole batch of measured data towards the base station.

.By the same token, another highly effective and reliable minimum energy compressed data aggregation scheme with CS is conferred in [11], which implements diffusion wavelets to discriminate the spatial correlations for reliable recovery of data.

From the analysis of the outcomes, it is revealed that highly reliable recovery of data is possible by the appropriate plotting of inadequate basis which is an energy efficient way for data gathering.

To achieve the goal of increasing performance, efficiency by the proper utilization of resource through various transitions optimization in ether end to end or on intermediate nodes in the constrained network.

## CHAPTER 4: PROPOSED APPROACH

Before coming up to the solution approach we had passed through a lot as discussed on the preceding chapters besides of considering the background and knowing the problem in order to fill the gap , we have attempt enough to review related works to have the flawless sight of how and where to go.

The goal of the proposed approach which is in-network approach of lossy aggregation is to optimizing multiple conditional temperature requests in to a single resource by merging multiple conditional observation and responding back to the client as a single resource is succeeded by having a CoAP server aggregating multiple CoAP observation values from the observers list on. More precisely, the server registers the observed temperature request messages only once for the same subject so that this work is targeted to make harmonious suit for the multiple temperature observers to fit in the suitable subjects. Aiming to reduce energy consumption and network traffic loads.

### 4.1 CONTRIBUTION

To get rid of unnecessary transitions and unwanted notifications to the clients, we used the new CoAP option ‘Condition’ as an extension of Observe Option aiming to support conditional observation. This condition option could be used by CoAP clients to set their interest of conditions. as our proposition, even if its widely expected that multiple observation request to takes place but only when the client Defined standard set of condition is meet, the CoAP server will send a notification response whenever state appear to be fresh or state changes. But when the CoAP observers fail to meet the set of standards that have been set to be observed or when condition is not met then the CoAP server is not expected to send notification response. IF-MATCH option needed in use to support by making the request conditional and for the resource update requests on the current existence. there is the highest possibility of IF-MATCH option to happen multiple times whenever any of the options match which means the condition is fulfilled but in contrary if there is no IF\_MATCH option and none of the option match then it clear condition is not fulfilled so the process should loop back checking for the state changes of new observers if they came up to be satisfying condition or not.

When the approach is successful then the following situation must be successful.

#### **4.1.1 Data Aggregation:**

it's defined as the process of aggregating the data from multiple sensors to eliminate redundant transmissions. The main objective is to create a data aggregation procedure as efficient as possible, using techniques of existing protocols, improving these techniques and improve different parts of the protocol.

#### **4.1.2 Energy Consumption:**

The main goal of data aggregation algorithms is to gather and aggregate data in an energy efficient manner so that network lifetime is enhanced. Whereupon the clear objective of creating a new data aggregation is trying to reduce as much as possible the traffic of redundant information, so reduce as much as possible energy consumption, try to reduce the traffic load and conserve energy of the sensors. better data aggregation and therefore reduce network energy consumption and extend the life of the nodes. These improve are:

- Minimize the amount traffic of the network.
- Minimize the energy consumption of the network.
- Increase the network life time.
- The Selection of which sensor node sends when redundant data is detected must be better balanced.

#### **4.1.3 Traffic Load reduction:**

the traffic should be minimized to minimize the load on the network and also for energy efficiency. While the application protocol CoAP traffic exchanged during on the period of time. The CoAP Client and Server triggers high amount of traffic load especially while the CoAP observers happened to be multiple.

#### **4.1.4 Save Memory:**

the procedure of IP based loads trigger challenges to tiny, constrained sensor nodes in terms of memory and processing power. When memory gets to be low then the lower is the better promoting low power consumption and low cost. Memory is biggest limitation on constrained devices challenging processing

capability, power, bandwidth so our proposition mainly targeted to reduce memory usage as a backbone of simplicity for the optimization of resources we needed to achieve as the highest priority.

#### 4.1.5 Costly Transmission

data transmission is really expensive so that lower transmission cost very essential in WSNs not to have disrupted network performances by preventing successful transmission of packets.

## 4.2 PROPOSED APPROACH

### 4.2.1 CONSTRAINED ENVIRONMENT

Observation aggregation is crucial paradigm for routing in wireless sensor networks. We will be performing by using function like average, max or min to aggregate multiple data coming from sources (client) to reduce communication redundancy and thus save energy as well as bandwidth to achieve proper utilization of network resources and maintain fairness among all sources. Previous sections explain the architecture of CoAP and how the different parts work together to fulfill the processing chain of CoAP messages. In the following section, a more detailed introduction to how a CoAP message is modeled and how these components can be implemented is given.



Figure 11: Architecture Of Constrained Environment Communication

Whenever nodes send multiple resources, they have to be optimized likely by aggregating them together into a single CoRE link format architecture. Aggregation is accountable for growing the network lifetime and decreasing the energy intake. However, every node has the ability to keep data received from the next nodes or data generated by itself for a period of time, aggregate them and send out the aggregated resource to get rid of and the communication rate is decreased. For the reasons of IoT nodes are typically resource limited and battery constrained so Data must be aggregated to keep resources and energy optimized well.

#### 4.2.2 PROPOSED FRAMEWORK

Having in mind the architecture of constrained environment communication, the proposed framework takes issues into account by averaging arriving packets in the server before the queue in the client's buffer overflows. The proposed architecture represents, if there is any kind of state variations happening on the server then it needs to check if there is established conditional relationships or not. After the condition is met the sever shall send the notification response back to the client that has established the conformed relationship. But if the condition failed to meet the condition, then the server is not expected to notify the observer.

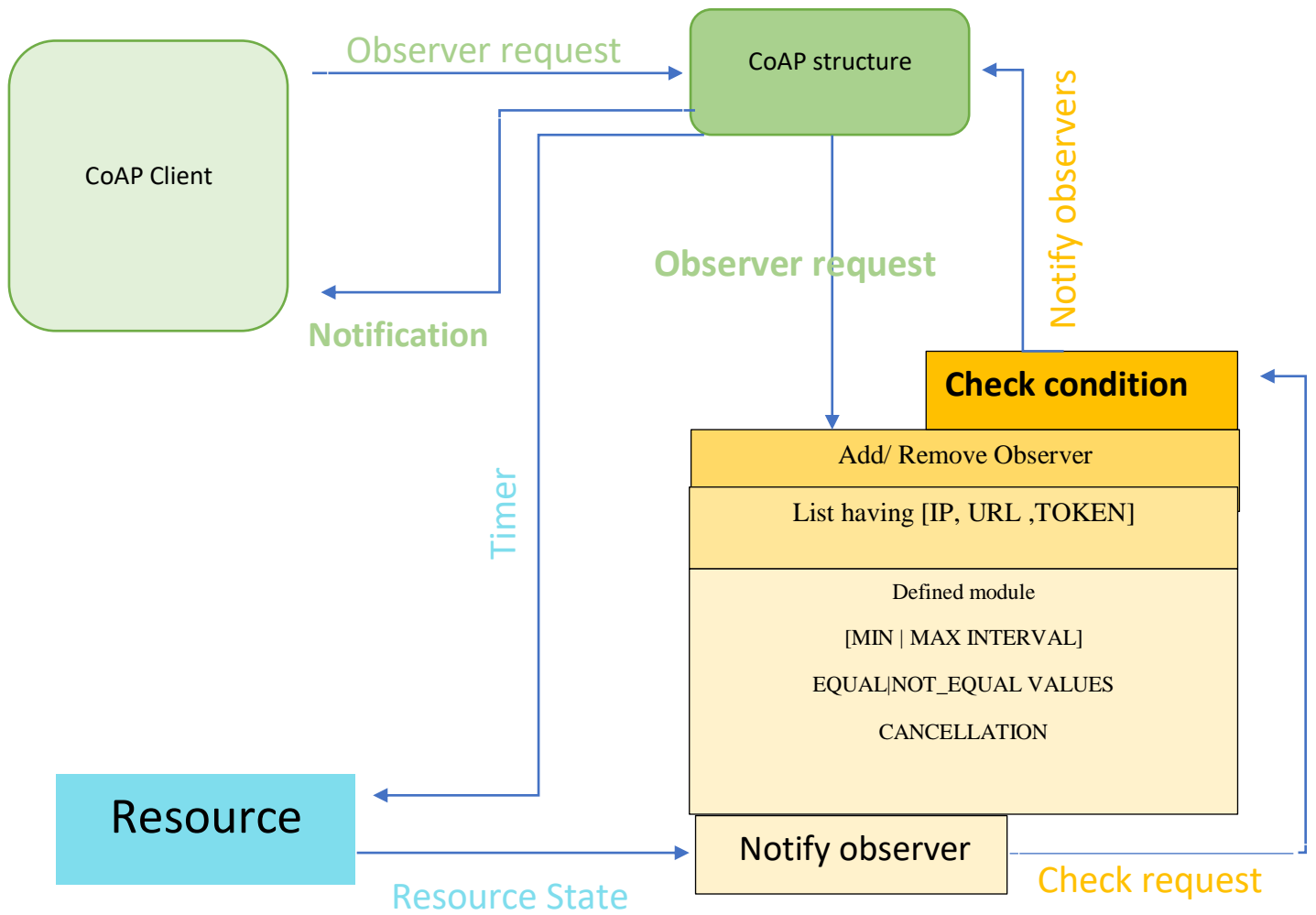


Figure 12: CoAP Observe conditionally, if condition met

#### 4.3 DETAILED APPROACH



#### 4.3.1 REGISTER AND NOTIFY CONDITIONAL OBSERVERS

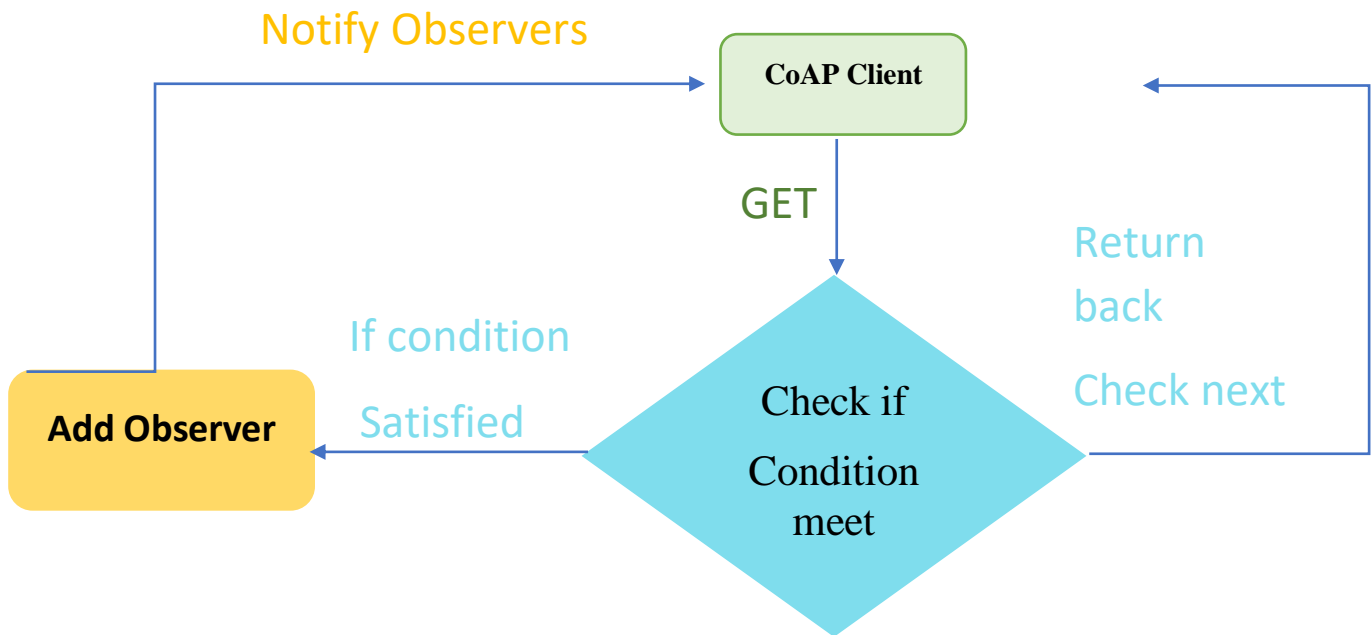


Figure 13 : Registration and Notification

When a client desires to initiate a conditional observation relationship, it sends GET request with observe having condition option. the option we refer as a condition accommodates set by observer clients having MIN |MAX response time, GRATER | LESS than values and more . after the server received the GET request then checks it if the requests satisfy the condition option by referring observers if URL and IP exist then if the CoAP client found to be fulfilling the condition options then the client will be notified the successful establishment of conditional relationship. but if not, the new observation request which is net in line will be checked.

Meanwhile, condition option is optional so that whenever CoAP client having observe request with condition option will automatically fall back to a basic observe request if the server could not provision the condition option. As data exchange between the source to endpoint there is a process to register, send or receive CoAP requests the conversation always starts by client showing up, upon the reception of request then upon the reception of request the endpoint notifies the observer through the UDP socket by sending a NOTIFY response.

Those kinds of notifications are not conversations rather they are one of the task endpoints to fetch the notified conditional observers when CoAP server is ready. The overall observation, registration and notification process are always ended by CoAP server response. the last acknowledgments must always last notifying observer that conditional observation is successful. But in situations like observation fail of end up to be not satisfying condition option which means if error occurs the end point still continues to check until satisfying conditions found server acknowledges successful.

#### 4.4 ADD CONDITIONAL OBSERVERS

Once the CoAP client observe issue initiated then it will be registered as an observer on observers list. while registering or adding the client as an observer requires `coap_add_observer` command with corresponding attributes of (URL, IP, TOKEN. Following section presents an algorithm determining whether to add CoAP client as observer if condition is well defined

- CoAP client initiates GET request in a resource
- If condition option defined
- Server take the client endpoint and token specified to the list of observers
- Add CoAP observer in to list of observers, else add normal observer;

##### Pseudo Code For Adding Additional Observer

```
1. if CONDITION Defined
2. Add_coap_obs [URL, Ip, port];
3. COAP_OBSERVING_REFRESH_INTERVAL
4. Add observer |observers_list|
5. Else
6. Add Normal observer;
```

##### Algorithm 1: Add Conditional Observer

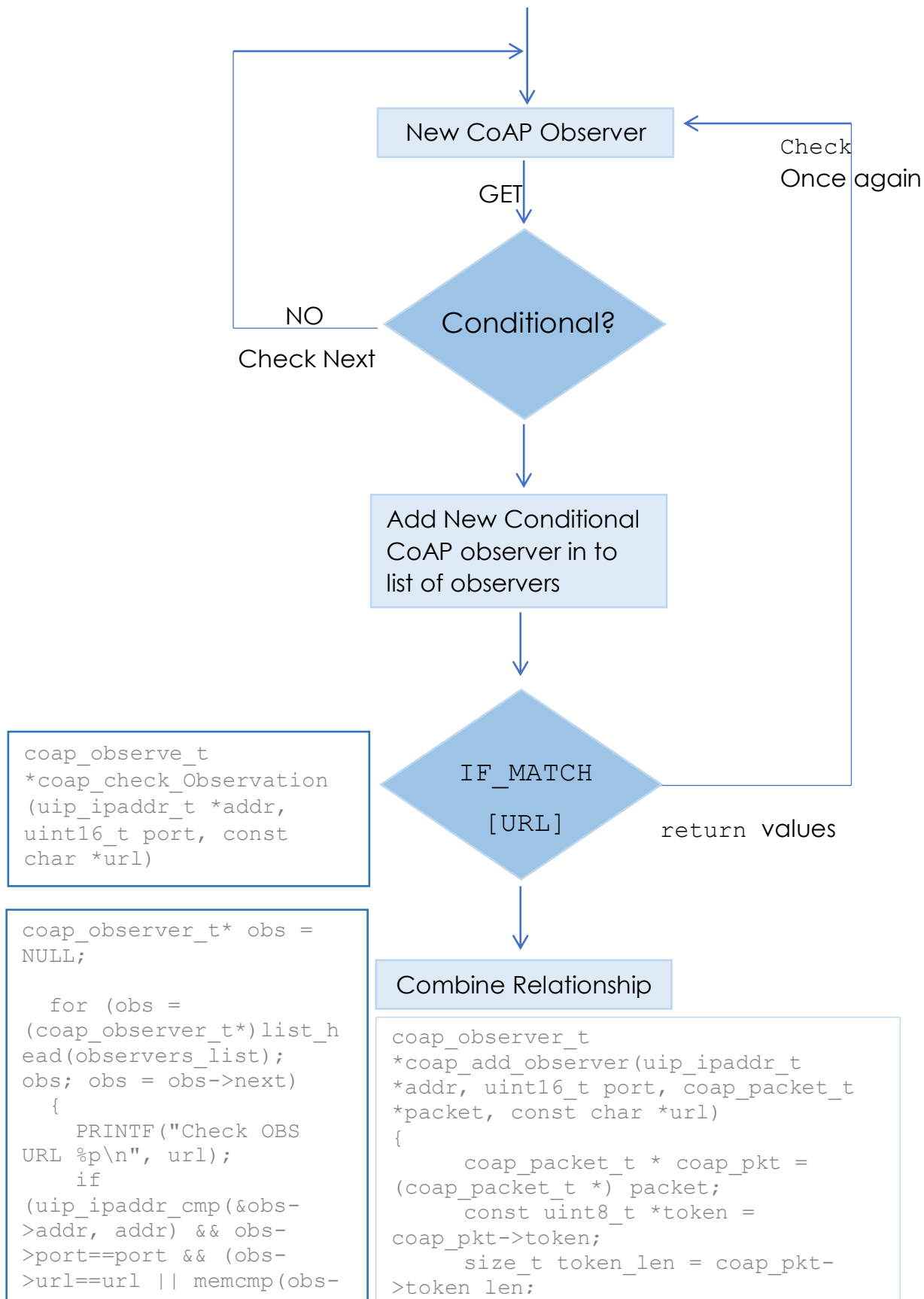


Fig 14. Add Observer functionality

## 4.5 REGISTER AND NOTIFY CONDITIONAL OBSERVERS

As the above pseudo describes, whenever client appear satisfying condition then it will be registered under CoAP list of observers but if the condition can't be fulfilled then the observer will be considered as normal observer. in our proposed solution after the observer added there should also be an observe function of allowing another client to send a request so that it could be registered in to list of observers. This is where we create new table for newly permitted CoAP client so in order to achieve this the new observer request must accommodate the URL,IP address and of course the destination address that receive data from the client while observing .after receiving that ,the new client will be registered as an observer .so the newly defined function which is the aggregator function make sures if observer from source to same resource occurs so if the result happened to be yes then we combine values by adding additional parameter to existing relationship but if the result returned to be no ,we mandatorily expected to add new relationship because we mainly aim to modify multiple observation request to be issued as a single observe. Finally, conditional CoAP observers that are located on observers table conditions will be compared to start combining condition satisfied values or to re priced from top until the return is successful. The following see how the implementation algorithm should go.

1. Check if the added CoAP observer has established relationship, which means
2. Check if CoAP observers matching URL
3. If match, combine multiple observers observe value(condition satisfied)
4. If no, then add new relationship (by returning back to checking the observers relationship)

### Pseudo Code For Registration And Notification

1. If CONDITION Defined
2. Add\_coap\_obs [URL, Ip, port];
3. COAP\_OBSERVING\_REFRESH\_INTERVAL
4. Add observer |observers\_list|
5. Coap\_check\_observation
6. Combine relationship |IF\_MATCH| then return values
7. return and check\_ once again

### Algorithm 2: Register And Notify Conditional Observer

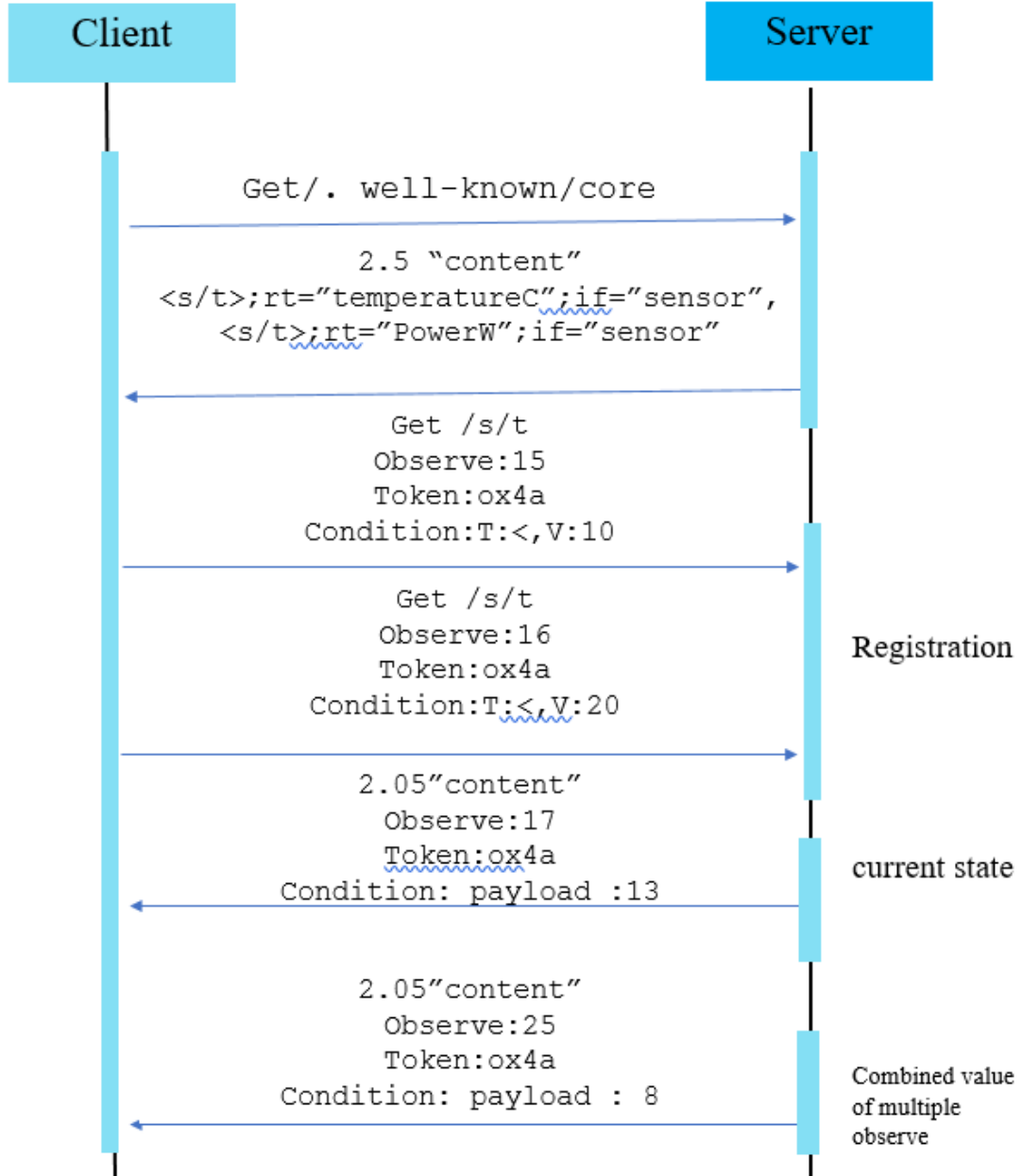


Figure 15: Proposed Conditional Observation work flow

Figure demonstrates conditional CoAP client registering its interest in a resource .while the state of resource variations as defined conditionally then the CoAP server notifies each client in the list of observers of the resource.

Each notification is an additional CoAP response sent by the server in reply to the GET request and includes a complete, updated representation of the new resource state.

## 4.6 AGGREGATOR FUNCTION

The aggregation function is the process of particular nodes that gather the collected results from other nodes. The gathered data must be controlled by the node to lessen the transmission, which is the base station or often an outer user who may have authorization to interact with the network. An aggregation process must have an aggregate function to compute the aggregated result from raw data. An aggregate function exhibits a set of characteristics that we interpret as features.

Aggregation involves combining the data that belong the same phenomenon. The main objective of data aggregation is to increase the network lifetime by reducing the resource consumption of sensor nodes (such as battery energy and bandwidth).[9] While increasing network lifetime, data aggregation protocols may degrade important quality of service metrics in wireless sensor networks, such as data accuracy, latency, fault-tolerance, and security. Therefore, the design of an efficient data aggregation protocol is an inherently challenging task because the protocol designer must tradeoff between energy efficiency, data accuracy, latency, fault-tolerance, and security. [18]In order to achieve this trade off, data aggregation techniques are tightly coupled with how packets are routed through the network. Hence, the architecture of the sensor network plays a vital role in the performance of different data aggregation protocols. There are several protocols that allow routing and aggregation of data packets simultaneously. The aggregation can solve this problem by averaging arriving packets in the server before the queue in the client's buffer overflows. Mostly, data aggregation approaches offer numerous architectures as follows. Data aggregation is the process where raw data is gathered and expressed in a summary form for statistical analysis. For example, raw data can be aggregated over a given time period to provide statistics such as average, minimum, maximum, sum, and count. After the data is aggregated and written to a view or report, you can analyse the aggregated data to gain insights about particular resources or resource groups. There are two types of data aggregation:

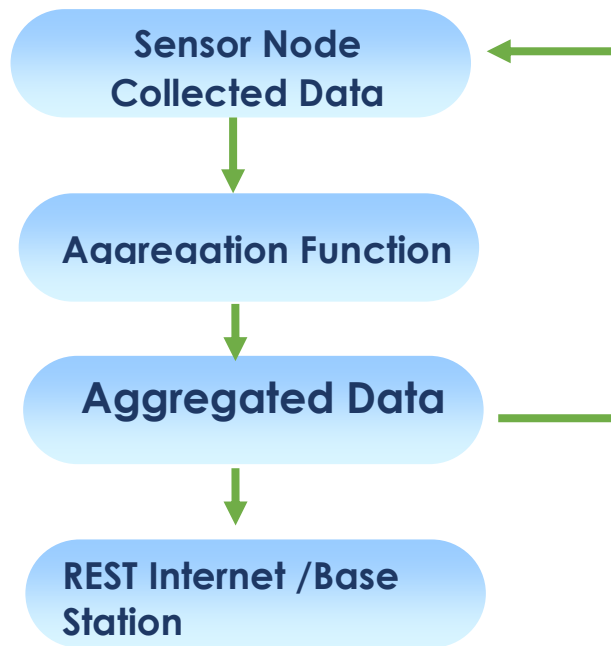


Figure 16: Architecture of aggregation function

*Sensor nodes* : are the foundations of information performing sensing as core task in addition to forwarding messages in the network as the condition of communication paradigm. *sensor nodes* are capable to self-organize to *gather* information about the atmosphere where they are positioned.

*Aggregation function*: aggregate function is a function where the values of multiple resource information together to form a single summary value or a single relationship. Common aggregate functions computes set of values and returns single value. How to aggregate is defined using aggregation function. This issue is the most important part for data aggregation, which focuses on how sensor nodes aggregate raw data into a digest. An efficient and useful aggregation function helps sensor nodes to reduce energy consumption dramatically. We have mentioned above that there are two correlations (i.e. temporal and spatial correlations) in raw data, and data aggregation function mainly benefits from these correlations. Basically, simple operations are used as aggregation functions (based on temporal correlation), such as Average, MAX(MIN), SUM, COUNT, and Median [19].

*Distinctive support nodes*: perform a specific task, such as acting as intermediate data collectors. They are not sources nor destinations of messages, but exploit mobility to support network operation like that of computing aggregation aggregated data:

*aggregator node* : act as a head of a cluster This node can be used to aggregate numeric values over a specific time span that combine more than one sensor inputs to one output (as a way to implement some sort of a simple sensor redundancy) aggregation of more than one inputs.

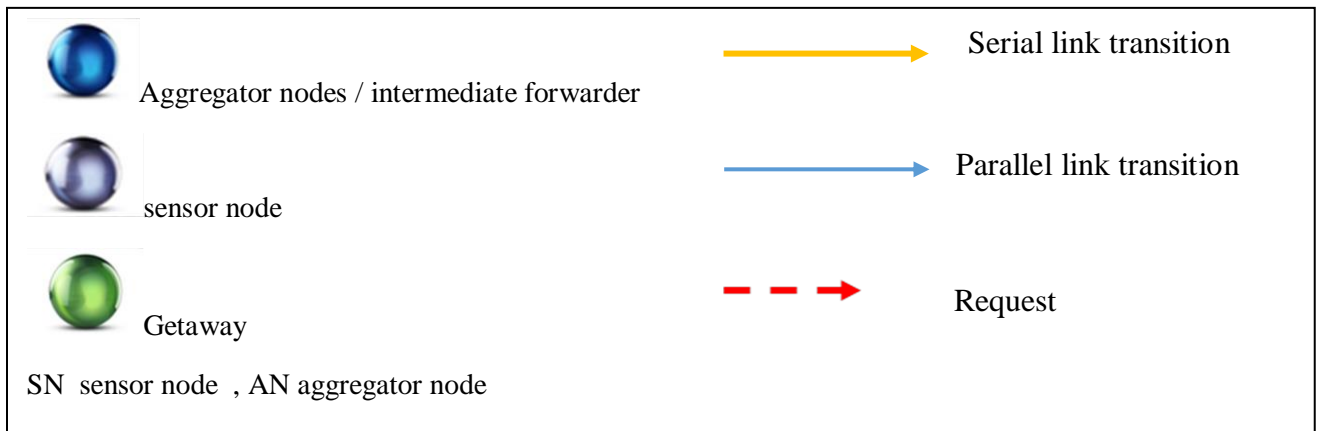
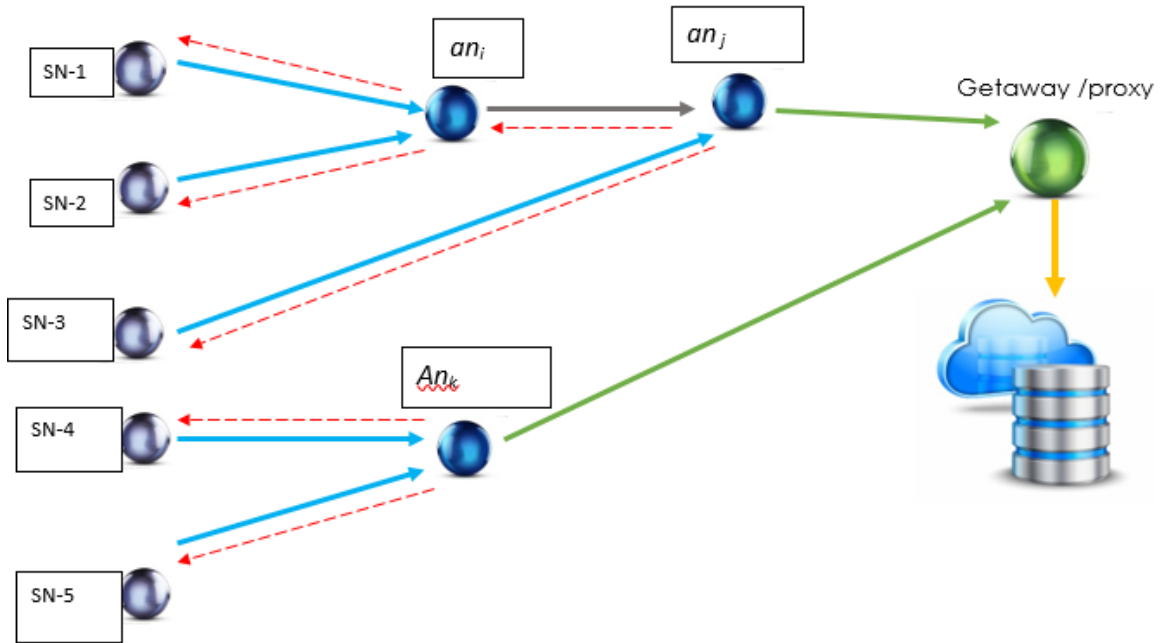


Figure 17: Process of aggregation

In order to accomplish data collecting in a heterogeneous sensor network, in-network processing paradigm can be implemented [18]. That is, a routing tree rooted at the base station and spanning aggregator nodes and sensors will be used for data collecting. The tree actually is a 2-tier cluster



routing tree, in which the aggregate nodes function as the cluster heads and the sensors attend as the members of clusters. All sensor node has only one cluster head, the sensor is a member of the cluster head. Within a cluster, each member sensor can forward its sensing data to the cluster head over multiple hop member sensor dispatch.

After the cluster head gathers all the sensing data from both its associates and its successor aggregate nodes, it then processes and transmits the collected data to its parent aggregate node. All data will be collected eventually through using the 2-tier cluster tree. Through single data gathering period, each sensor will consume the same amount of transmission energy by transmitting the same length message, since they also have matching transmission ranges. Yet, the transmission energy consumption of different aggregate nodes by transmitting a unit-length message is numerous.

As the above figure of process diagram of data aggregation demonstrates that the aggregate node  $an_x$  gets its energy by transmitting data for its successor aggregate nodes. So, the lifetime of the routing tree consisting of aggregate nodes will define the lifetime of the heterogeneous sensor network. Precisely, let  $an_x, an_y, \dots, an_p$  be the  $P$  aggregate nodes with each  $an_x$  energy intake  $ei(an_x)$  per data gathering,  $1 \leq x \leq P$ .

Let  $T(n, P)$  be the network lifetime of the resulting heterogeneous sensor network by placing the  $P$  aggregate nodes into a sensor network, so that

$$T(n, P) = \frac{EC_{agg}}{\max_{1 \leq x \leq P} \{ei(an_x)\}} \quad \text{Eq (1)}$$

So that  $EC_{agg}$  is the energy capacity of aggregate nodes. Eq (1) shows that the network lifetime is inversely proportional to the maximum energy consumption among the aggregate nodes. So this leads us to put minimizing the maximum energy consumption among the aggregate nodes as a higher priority throughout this paper.

## 4.7 AGGREGATION FUNCTION MATHEMATICAL MODEL

Basic Aggregation functions are simple processes.

**1<sup>st</sup>.** Average, its effortlessness to implement on sensor node. Let raw data is represented by  $d_t$  at a time  $t$ , represented as  $(d_t, t)$ . the function of average can get an combined assessment by means of:

$$f_{average} \equiv \bar{d} = \frac{d_1 + \dots + d_t}{t} \quad \text{Eq (2)}$$

$\bar{d}$  represents the average price of  $t$  data.

**2<sup>nd</sup>.** Where ,MAX (*resp.* MIN), maximum (*resp.* minimum) is essential for alarming applications. Those applications don't have to know data, but if the data is too high (*resp.* too low) for their necessities, they need to activate alert. Function of MAX (*resp.* MIN) can be

$$F_{max} \equiv d_{max} = \text{MAX} \{d_0, \dots, d_t\} \quad \text{Eq (3)}$$

$$F_{min} \equiv d_{min} = \text{MIN} \{d_0, \dots, d_t\}$$

$d_{max}$  (*resp.*  $d_{min}$ ) indicates the MAX (*resp.* MIN) value of earlier  $t$  data.

**3<sup>rd</sup>.** The summation of the data  $d$  is represented by SUM and computing will be made as:

$$f_{sum} \equiv d_{sum} (t) = \sum_{i=1}^t d_i \quad \text{Eq (4)}$$

$d_{sum}$  have sum value of  $\{d_1, \dots, d_t\}$ . Likewise procedures like COUNT (that we use to count number of data collected), Median which is collected median value.

The problems of those tasks are that they will not promise the accurateness according to a given error threshold. Application can choose one of them to achieve energy efficiency and network capacity saving if accuracy is not a hard requirement.

## 4.8 PROPOSED APPROACHES PHASES

### Phase 1: Harvesting Node

- pick out proper of modules to modify and the right node to set up
- The right selection of node with their processing module is so essential to accomplish effective energy consumption.

## Phase 2: Application Module Defining And Executing Transactions

- To launch a process and excite defined transactions on potential node
- set the **module** to generate an event on a defined period
- Logs and modification of scripted module applied on Harvested node.

## Phase 3: Defining Condition Option

- avoid transmission of unwanted notifications to clients
- require some additional processing for checking the condition
- Condition Option represents the condition the client wants to apply to the observation relationship
- used to describe the resource states the client is interested in.
- notifications will be sent only when the resource state meets the condition specified in the Condition Option

## Phase 4: Sensors Node Alliance

- After the application module dynamically loaded and linkage established
- Sensor nodes must be alliancing with aggregator nodes
- Achieved by using **CoAP Observe**
- **CoAP** contain normalized **URIs** with their path and **query** split into multiple segments.
- **GET** method retrieves a representation for the info that currently corresponds to the resource identified by request URL

## Phase 5: Packet data transmission

- after dynamic loading then the packet data interchange begins
- The regular observe function modified to enable to add additional observer in to the observers list by
  - Sending observation request to sensor node
  - Request it to register the aggregator as an observer rather than itself.
  - alliance of resource on a sensor to a aggregator node will be successful when the new added observer have (url, IP, path)
  - sensor node registers the aggregator as an observer and in our case

## CHAPTER 5: IMPLEMENTATION AND RESULTS

Aggregation of packets the most essential approach for resource management and energy efficient data gathering. When we perform this using Contiki platform, trial results and measured outcomes can be acquired using numerous methods and approaches. On this thesis work, we made couple of trials and execution of proposed arrangements.

### 5.1 Experiment Setup

To simulate the network and evaluate our congestion control mechanism, the algorithms were implemented using the Cooja Simulation in the Contiki Operating System (Contiki OS), an open source operating system for constrained devices that supports dynamic loading and the replacement of individual programs and services. The advantage of Cooja was that the program can compile and upload into a real node. In this study, we selected a Sky mote which was developed by to be the principal node for simulations in the Cooja [29]. It was devised for a low-power wireless sensor network (WSN) which can communicate over the 6LoWPAN protocol.

**Traffic Scenarios** Two traffic scenarios were set to investigate the effects of the two congestion control mechanisms on the performance of the Obs communications. **Continuous traffic:** The CoAP server delivered the CON request notifications to the CoAP client. As the server obtained a reply from the client, it immediately transferred another CON request. Sending messages back-to-back from a large number of servers at the same time can create congestion. In this study, we used nine servers in order to achieve different levels of congestion. The test was performed continuously for 300 s and was repeated 20 times for each specific configuration. [7] **Burst traffic:** This scenario simulated the burst traffic by increasing the congestion level in every topology. It started with a low congestion level at the server, where four nodes generated the continuous traffic (Obs) of back-to-back CON requests. Afterwards, a burst of traffic was generated by another group of servers consisting of the rest of the servers in the topology. For instance, network topology I had a group of three servers for burst traffic, i.e., three nodes, in order to achieve different levels of congestion.

The test in this scenario was done in a similar manner to the continuous traffic scenario. The aggregation process defines how the resources are joined together and thus the final energy scheduled per group. The aggregator can then use this energy in an auction in energy markets, giving the possibility of negotiation at the level of larger resources when comparing with individual consumers or distributed generators.

### **5.1.1 Contiki OS**

The proposed approach is implemented by the help of Contiki operating system. Contiki is such a platform which is developed to afford this liberty to execute dynamic loading and unloading of codes smoothly. Contiki is established with an event-driven kernel which accommodates selective preventive multi-threading for each exclusive process. It is established on C language which supports multiple environments to configure several micro controller architectures containing MSP430, Atmel AVR, now based on ESB platform. The ESB adopts MSP430 with 2kb RAM and 60 kb ROM, operating at 1 MHz. The MSP430 is capable of performing optional reprogramming of on-chip flash storage. Memory allocation is an efficient way to provide a mechanism for memory allocation. with full IP networking affords a full IP network stack, with standard IP protocols such as UDP, TCP and HTTP of dynamic module loading dynamic loading and linking of modules at run-time is supported.

Power awareness operates in extremely low power systems which are designed to run for years on a pair of AA batteries. The Cooja network simulator makes simulation tremendously easier by providing a simulation environment and the Rime Stack supports simple operations such as sending a message to all neighbors or to a specified neighbor, as well as more complex mechanisms such as network flooding and address free multi-hop semi-reliable scalable data collection, The Contiki shell provides an optional command line shell with a set of commands that are useful during development and debugging of Contiki systems. Protothreads is a mixture of the event driven and the multi-threaded programming mechanisms. With Protothreads, event handlers can be made to block, waiting for events to occur.

## 5.2 Contiki Communication

Contiki supports standard protocols and recent enabling protocols for IoT –

- **uIP (for IPv4)** – This TCP/IP implementation supports 8-bit and 16-bit microcontrollers.
- **uIPv6 (for IPv6)** – This is a fully compliant IPv6 extension to uIP.
- **Rime** – This alternative stack provides a solution when IPv4 or IPv6 prove prohibitive. It offers a set of primitives for low-power systems.
- **6LoWPAN** – This stands for IPv6 over low-power wireless personal area networks. It provides compression technology to support the low data rate wireless needed by devices with limited resources.
- **RPL** – This distance vector IPv6 protocol for LLNs (low-power and lossy networks) allows the best possible path to be found in a complex network of devices with varied capability.
- **CoAP** – This protocol supports communication for simple devices, typically devices requiring heavy remote supervision.

### 5.2.1 Sky Mote

The Sky mote is the most simple of motes for use within a WSN and ideal for initial configurations within a Cooja simulation. We used Sky mote type because it supports 6LoWPAN, which is an advanced version of IPv6; Sky mote has 4 MHz clock speed. It is the difference between the real wireless sensor module and the module that is emulated in Cooja. It is important for power consumption estimation of Cooja WSN nodes. Uses the Texas Instruments MSP430F1611 [16] and MICAz relies on the Atmel Atmega128 [31], but both include the same wireless transceiver. Sky mote platforms, with the total size of the final executable files being the ones it should be noticed that not all the applications could be compiled for all the platforms.

- CC2420 RF Chip, MSP430F1611 Micro controller. IEEE 802.15.4 WSN
- Memory: 48 KB ROM
- Integrated peripherals introducing a 12 bit ADC (Analog to Digital Converter) and DAC (Digital to Analog converter) timer, 12c spi(serial peripheral interface) and UART bus protocols.
- Sky mote Platform Measuring of relative humidity, temperature and high via sensors.
- Inter-probability with many different IEEE 802.15.4 devices
- Very low current consumption and fast weak up from sleep
- The implementation of communication and mesh networking

### 5.2.2 Cooja Network Simulator

Cooja Simulator is a network simulator specifically designed for Wireless Sensor Networks. A summary of how Cooja executes binaries of different platforms. is also the default simulator of the well-known Contiki Operating system. Contiki implements IP for constrained nodes. If you are willing to test new protocols for IP\_enabled nodes, Cooja is a better choice than NS2 or OMNET. Several research results have been published based on Cooja and Contiki . this Contiki network simulator. Cooja allows the large and small networks of Contiki motes to be simulated. This article takes the reader through the process of programming IoT with Contiki and Cooja.[22]

Here we briefly describe the functionalities of each tool:

- **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, log outputs, etc. Initially this window is empty and we need to populate it with our sensors.
- **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.
- **Notes** - This is a simple notepad for taking notes about the simulation.
- **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.
- **Timeline** - Simulation timeline where messages and events such as channel change, LEDs change, log outputs, etc are shown.

In addition to the default tools, it is possible to exhibit other tools such as **Breakpoints**, **Radio messages**, **Script editor**, **Buffer view** and **Mote duty cycle**, which can be enable in the *Tools* menu

### 5.2.3 Erbium

#### EXAMPLE FILES

- `er-example-server.c`: A RESTful server example showing how to use the REST layer to develop server-side applications (for now CoAP is implemented for the REST Engine).
- `er-example-client.c`: A CoAP client that polls the `/actuators/toggle` resource every 10 seconds and cycles through 4 resources (target address is hard-coded).
- `er-plugtest-server.c`: The server used for draft compliance testing at ETSI IoT CoAP Plugtests. Erbium (Er) participated in Paris, France, March 2012 and Sophia-Anti polis, France, November 2012 (configured for minimal-net).

### 5.2.4 Preliminaries

- Make sure `rpl-border-router` has the same stack and fits into mote memory: You can disable in `border-router project-conf.h` (not really required as BR keeps radio turned on).  
`#undef NETSTACK_CONF_RDC #define NETSTACK_CONF_RDC nullrdc_driver`
- Alternatively, you can use the `native-border-router` together with the `slip-radio`.
- Optional: Save your target as default target make `TARGET=sky` save target

#### DETAILS

Erbium implements the Proposed Standard of CoAP. Central features are commented in `er-example-server.c`. In general, `apps/er-coap` supports:

- CON Retransmissions (note `COAP_MAX_OPEN_TRANSACTIONS`)
- Blockwise Transfers (note `REST_MAX_CHUNK_SIZE`, see `er-plugtest-server.c` for Block1 uploads)
- Separate Responses (no `rest_set_pre_handler()` required anymore, note `coap_separate_accept()`, `_reject()`, and `_resume()`)
- Resource Discovery



- Observing Resources (see EVENT\_ and PRERIODIC\_RESOURCE, note COAP\_MAX\_OBSERVERS)

### 5.3 SIMULATION SETUP

This section shows the experiment setup and evaluation results, the simulation is designed as a single interface for the application of multiple simulation types. After we have chosen one border router and servers with multiple clients. The first step we need to do is choose the proper nodes for client, server and the gateway using sky mote, the simulation on the figure shows the necessary network setup before starting to run. Among the setup client sky nodes all of them are connected to the aggregator server that's connected to the border router for the clients to sense either temperature or humidity from the environment and send it to the aggregator server if the value fulfills the defined conditions then the server handle the incoming requests for combining the requests having relationship and registering as a single value followed by notify back for the representative nodes. To achieve this we have used VMware virtual work station player with instant contiki 2.7 operating system and Cooja as simulation.

In our experiment setup and evaluation results we needed to show the approaches we used to aggregate multiple packets and show how energy, memory and network load reduced compared to not aggregate CoAP clients observation and this carried out in Cooja emulator by using the sky nodes. However, efficiency and energy savings are evaluated experimentally showing that aggregation have very high and major performance, energy saving advantages. The residual energy of the node is taken for choosing the aggregator server so that successful aggregation is done at the CoAP server, and finally, it is transmitted to border router. As mentioned previously, the mockups are done using the COOJA simulator, and it is compared in terms of network lifetime and storage utilization tested in Hybrid and Tree topological structure of a network.

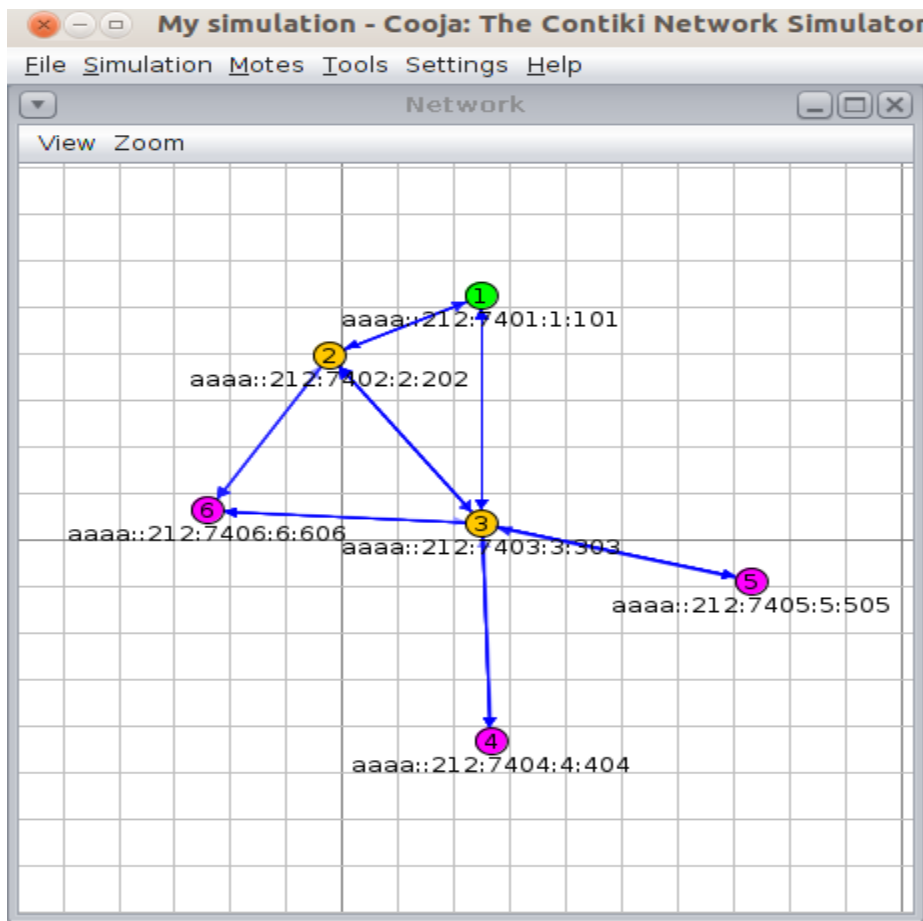


Figure 18: Simulation Setup

### 5.3.1 Simulation Setup Structure

Our nodes identified by the range of 1 to 6 appear to be representing rpl-border-router [1] and the rest are erbium records source module of client and servers accommodating loader of well conditionally defined application module in order the server to allow additional observer to establish relationship with the existing on a list of observers. Both servers [2] and [3] act as a head of the cluster to border transmission to gateway and handle by checking CoAP observers interest with merged multiple sensor values as a single relationship. Even if Cooja simulators sensors are not able to get information from the environment, we have built in sensors that our sensor or client nodes represented by (4,5,6) to generate data showed in er-server / client .c

## 5.4 Evaluation

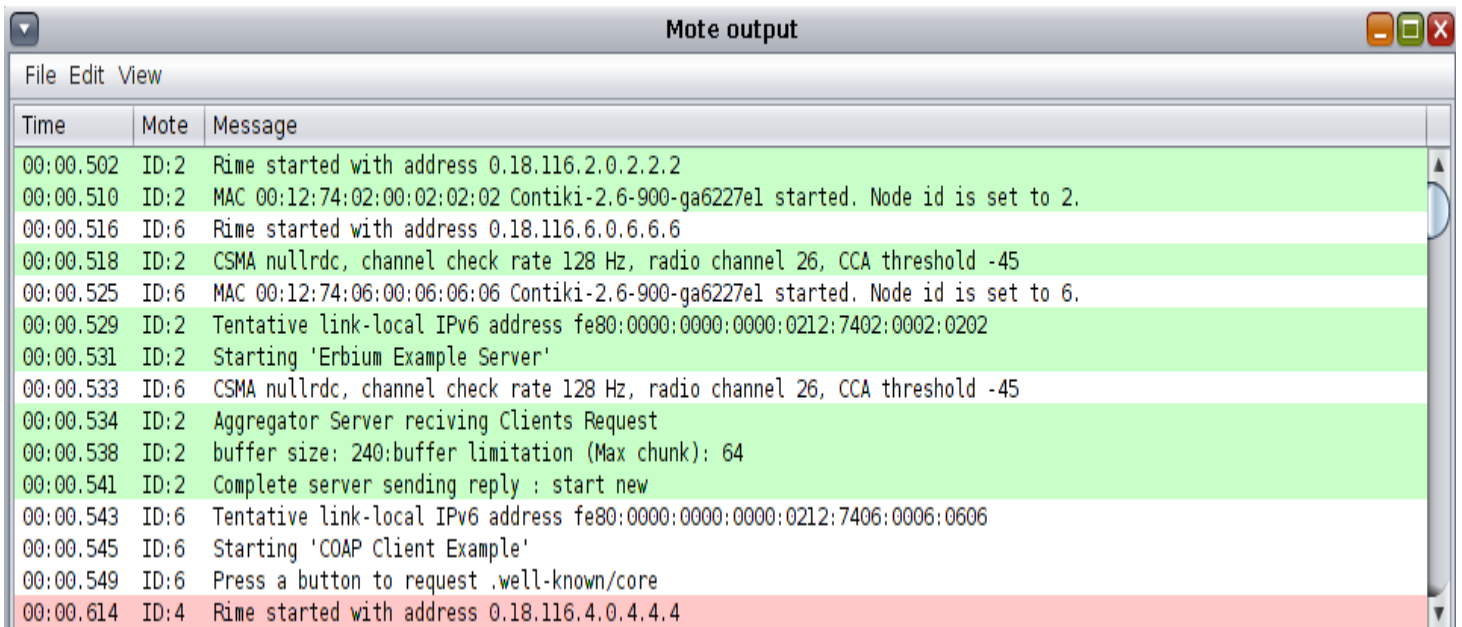
The CoAP prototype implementation is used to assess the proposed structure for scalable and consistent IoT services. are achieved against the state-of-the-art CoAP implementation for constrained environments. Valuation scenarios, however, can depend on the many different communication models in IoT applications, In our condition, the clients must register theme selves thereby issuing a request and having state changes when they update. So to better evaluate if the objectives have meet of the proposed implementation. However it allows us to make a compression with the original work

### 5.4.1 Functional evaluation

Cooja simulator offers diverse sensor nodes for simulations. In our evaluation Sky notes are used to perform simulations and performance of various Functions to be analysed

#### 5.4.1.1 Establishing the groundwork

Weather establishing the network or starting chipper or costly hand shake requires the condition al CoAP clients to make the first move of requesting for the resource of CoAP server for further aggregation. Our observation starts by the nearest node value that is represented by mote ID (4) after the introduction has been made.



Time	Mote	Message
00:00.502	ID:2	Rime started with address 0.18.116.2.0.2.2.2
00:00.510	ID:2	MAC 00:12:74:02:00:02:02:02 Contiki-2.6-900-ga6227e1 started. Node id is set to 2.
00:00.516	ID:6	Rime started with address 0.18.116.6.0.6.6.6
00:00.518	ID:2	CSMA nullrdc, channel check rate 128 Hz, radio channel 26, CCA threshold -45
00:00.525	ID:6	MAC 00:12:74:06:00:06:06:06 Contiki-2.6-900-ga6227e1 started. Node id is set to 6.
00:00.529	ID:2	Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7402:0002:0202
00:00.531	ID:2	Starting 'Erbium Example Server'
00:00.533	ID:6	CSMA nullrdc, channel check rate 128 Hz, radio channel 26, CCA threshold -45
00:00.534	ID:2	Aggregator Server reciving Clients Request
00:00.538	ID:2	buffer size: 240:buffer limitation (Max chunk): 64
00:00.541	ID:2	Complete server sending reply : start new
00:00.543	ID:6	Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7406:0006:0606
00:00.545	ID:6	Starting 'COAP Client Example'
00:00.549	ID:6	Press a button to request .well-known/core
00:00.614	ID:4	Rime started with address 0.18.116.4.0.4.4.4

Figure 19: Establish Network

### 5.4.1.2 Aggregator server

Cooja network simulator network id represented by ID 2 and ID 3 which are the intermediate or head of the cluster are the aggregator server, ID 3 takes sensor value of node ID 4 and 5 with IP aaaa::212:7405:5:505 and IP aaaa::212:7404:4:404 with Conditional CoAP client TCP source port number of 5683 .every incoming package from client nodes will be tested if MAX –AGE not expired thane handle requests after adding on observers list if the data did not pass the buffer limitation which is 64 and complete server sending and start new.

The screenshot displays the Cooja network simulator interface. On the left, a network diagram shows six nodes (1-6) connected in a mesh. Node 1 is at the top, node 2 is below it, node 3 is to the right of node 2, node 4 is below node 3, node 5 is to the right of node 3, and node 6 is to the left of node 3. Each node has an IPv6 address: Node 1: aaaa::212:7401:1:101, Node 2: aaaa::212:7402:2:202, Node 3: aaaa::212:7403:3:303, Node 4: aaaa::212:7404:4:404, Node 5: aaaa::212:7405:5:505, Node 6: aaaa::212:7406:6:606. On the right, a 'Mote output' window shows a log of messages for nodes 4, 1, 5, and 6. The messages include Rime startup information, MAC addresses, CSMA nullrdc parameters, tentative link-local IPv6 addresses, and starting of CoAP client and server processes.

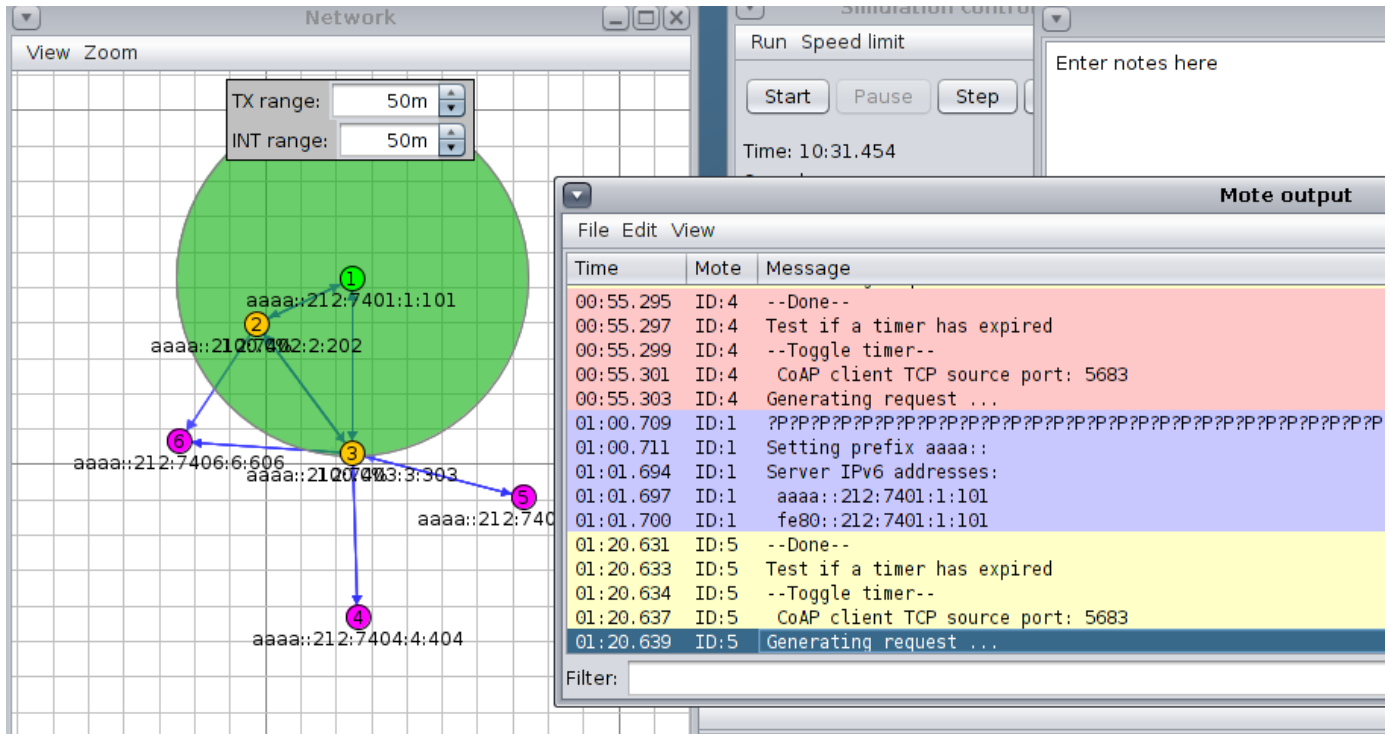
Time	Mote	Message
00:00.614	ID:4	Rime started with address 0.18.116.4.0.4.4.4
00:00.623	ID:4	MAC 00:12:74:04:00:04:04:04 Contiki-2.6-900-ga6227e1 started. Node id is set to 4.
00:00.631	ID:4	CSMA nullrdc, channel check rate 128 Hz, radio channel 26, CCA threshold -45
00:00.641	ID:4	Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7404:0004:0404
00:00.643	ID:4	Starting 'COAP Client Example'
00:00.646	ID:4	Press a button to request .well-known/core
00:00.659	ID:1	Rime started with address 0.18.116.1.0.1.1.1
00:00.668	ID:1	MAC 00:12:74:01:00:01:01:01 Contiki-2.6-900-ga6227e1 started. Node id is set to 1.
00:00.677	ID:1	CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26, CCA threshold -45
00:00.688	ID:1	Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7401:0001:0101
00:00.692	ID:1	Starting 'Border router process' 'Web server'
00:00.981	ID:5	Rime started with address 0.18.116.5.0.5.5.5
00:00.989	ID:5	MAC 00:12:74:05:00:05:05:05 Contiki-2.6-900-ga6227e1 started. Node id is set to 5.
00:00.997	ID:5	CSMA nullrdc, channel check rate 128 Hz, radio channel 26, CCA threshold -45
00:01.008	ID:5	Tentative link-local IPv6 address fe80:0000:0000:0000:0212:7405:0005:0505

Figure 20: Aggregator Server

### 5.4.1.3 Border router

Our `rpl-border-router` having mote ID:1 help us to connect one to the other network, routing our aggregated data between RPL network to the external network takes establishing the connection with the internal.

Figure 21 : Border Router



### 5.4.1.4 From Cooper

The following demonstration shows routing and neighbor against the time showed in cooper server and verified address results of border router if it has been set by using ping `aaaa::212:7401:1:101`

```
64 bytes from aaaa::212:7401:1:101: icmp_seq=156 ttl=64 time=6.62 ms
64 bytes from aaaa::212:7401:1:101: icmp_seq=157 ttl=64 time=3.42 ms
64 bytes from aaaa::212:7401:1:101: icmp_seq=158 ttl=64 time=43.4 ms
64 bytes from aaaa::212:7401:1:101: icmp_seq=159 ttl=64 time=8.18 ms
64 bytes from aaaa::212:7401:1:101: icmp_seq=160 ttl=64 time=10.9 ms
```

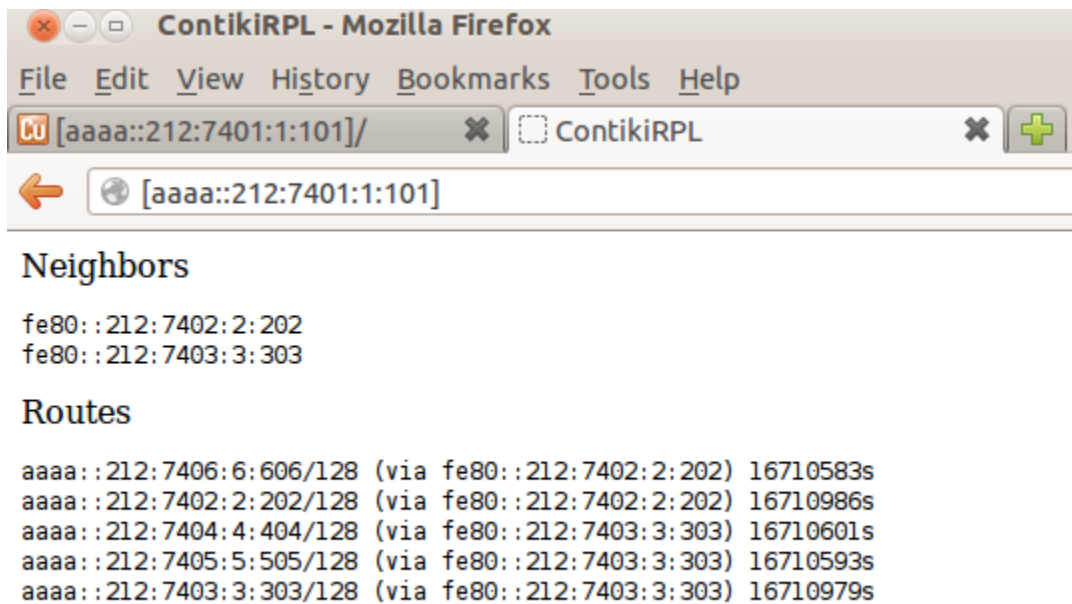


Figure 22: Routing Table

#### 5.4.1.5 Client Mots

The mote id represented by ID range of 4,5,6 will finally starts observation after it has been confirmed that it can be conditional observer then clients request and get notified the compressed value of single relationship.

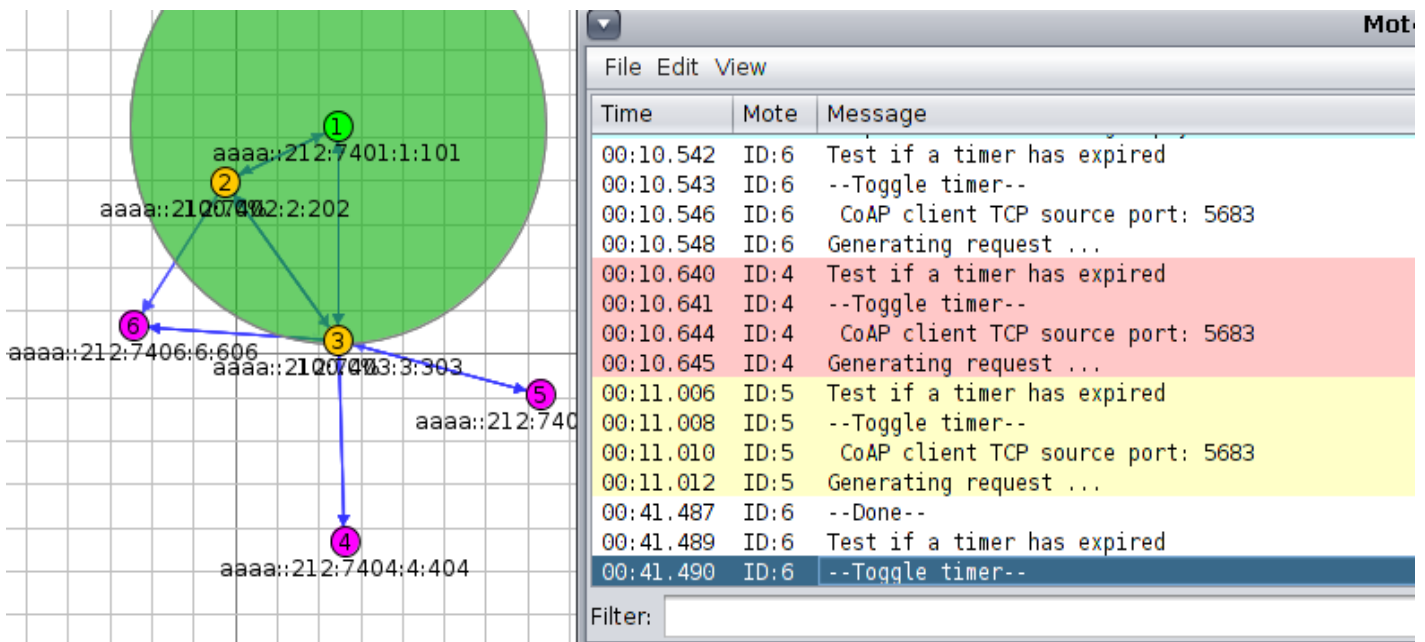
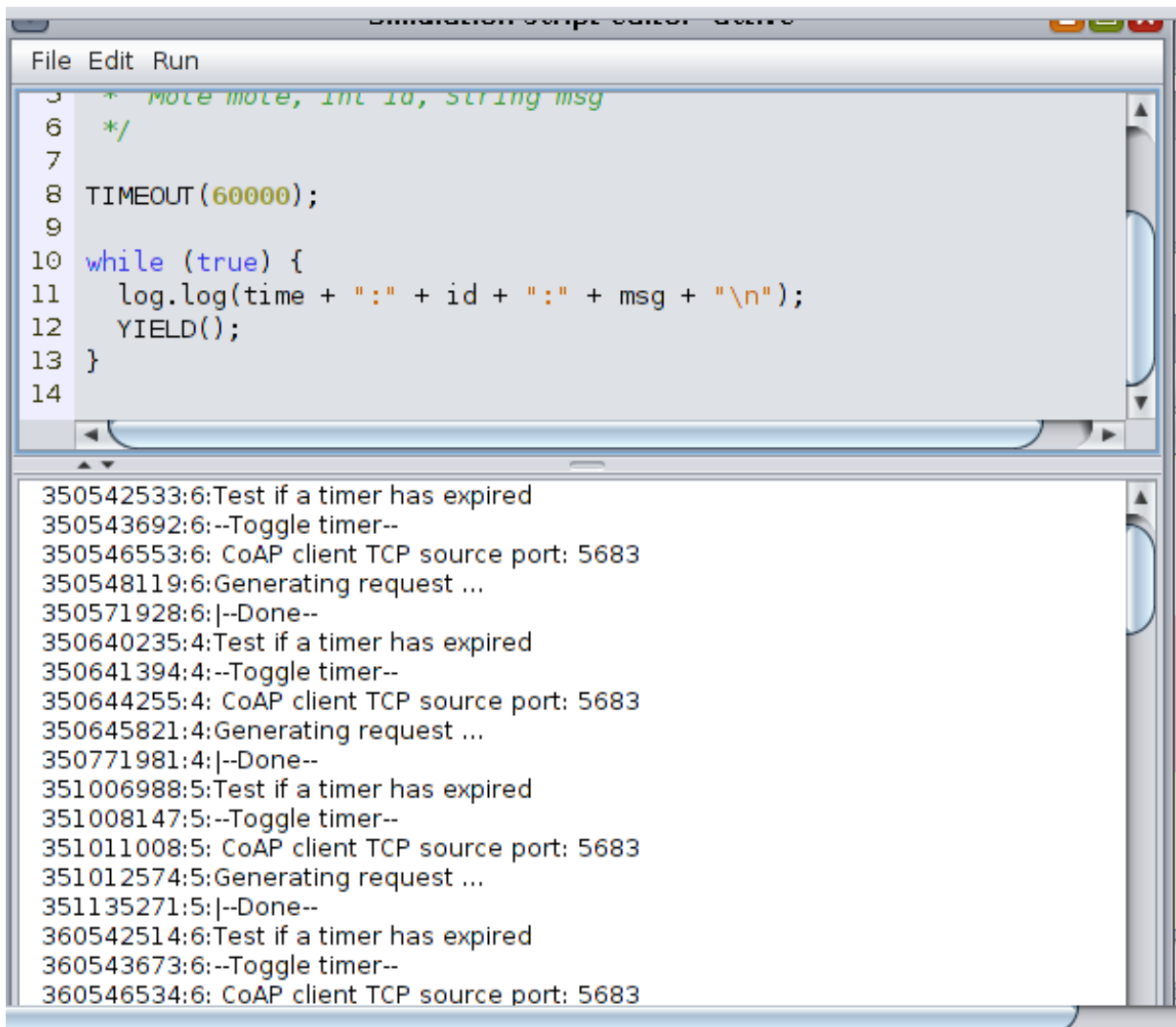


Figure 23: Client nodes output



```
File Edit Run
3  * mote mote, int id, String msg
6  */
7
8  TIMEOUT(60000);
9
10 while (true) {
11     log.log(time + ":" + id + ":" + msg + "\n");
12     YIELD();
13 }
14
```

```
350542533:6:Test if a timer has expired
350543692:6:--Toggle timer--
350546553:6: CoAP client TCP source port: 5683
350548119:6:Generating request ...
350571928:6:|--Done--
350640235:4:Test if a timer has expired
350641394:4:--Toggle timer--
350644255:4: CoAP client TCP source port: 5683
350645821:4:Generating request ...
350771981:4:|--Done--
351006988:5:Test if a timer has expired
351008147:5:--Toggle timer--
351011008:5: CoAP client TCP source port: 5683
351012574:5:Generating request ...
351135271:5:|--Done--
360542514:6:Test if a timer has expired
360543673:6:--Toggle timer--
360546534:6: CoAP client TCP source port: 5683
```

Figure 24: Clients output: script editor

#### 5.4.1.6 Radio traffic

In our experimentation, we use Cooja traffic analyzer to capture the radio message and save as the packet capture . Capturing the network traffic was done by collecting received packet data from all of the nodes running in the simulator to provide the collected view of data. As figure 22 demonstrates the use of Cooja traffic analyzer to capture the radio message and save as the packet capture showed in the picture



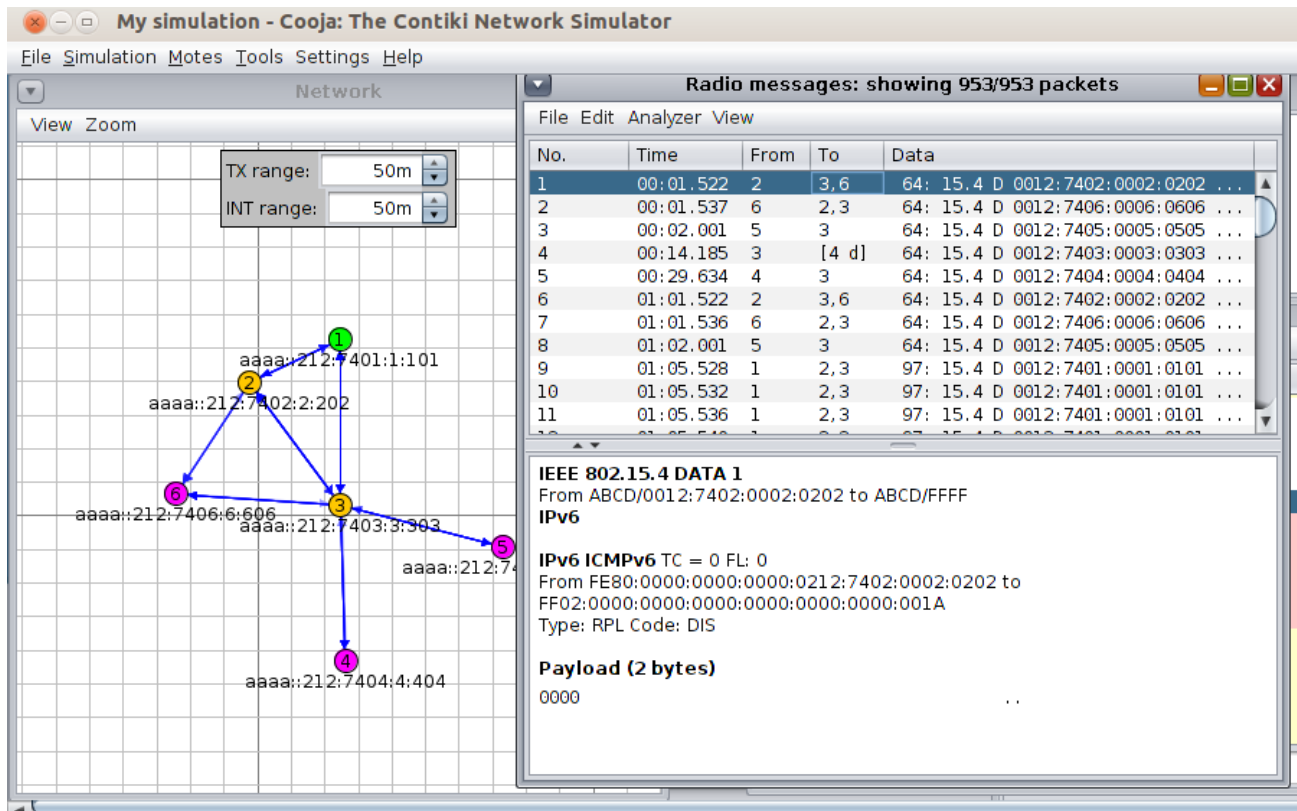


Figure 25: Radio Messages Enabled in Cooja simulator

Our network traffic is equated by radio message of packets against time and iteration in number of nodes. as Figure 22 demonstrates Radio Message Of 953 Packets or the amount of data that is moving across the network established between nodes at any given time. We can also call network traffic the data traffic, so as the name implies the data traffic can be broken down in to data packets then send them over a network before being reassembled by the receiving sky nodes.

## 5.5 Mathematical Evaluation

This performance evaluation tool aims to improve the performance of the end-point application. The total system performance doesn't always interrelate with the performance of the network path. Therefore, the network system performance should be evaluated in the client applications. Monitoring is the process used to track, evaluate and diagnose the performance of a network. With the variety of devices, Bandwidth commonly measured in bits/second is the maximum rate that information can be transferred.



### 5.5.1 Packets Records

In this section we analyze the network performance of a established hops' with number of nodes comparison of number of packets from sensor nodes to the base station. As the Figure demonstrates applying the aggregation function through multiple conditional observation process of packet numbers on established hops by competing average value. On our experiment, we used sky nodes pause as client and server so that when clients make request of multiple packets then the server takes them all by their relationship and compute the average value of the incoming packets checking if they have a matching relationship with the existing one. Let the average value of obs1, obs2, obs3 be Obs X for every incoming packet established a relationship with Obs X they are considered to be as a single relationship.

$$OBS = (obs1, obs2, obs3)$$

$$OBS_{avg} = \sum \frac{Obs1..Obs3..}{Obsn}$$

Where value of  $OBS_{Avg}$  will be notified back to the observers as a single package if only  $OBS_{avg}$  comes before MAG-AGE expires.

### 5.5.2 Power Consumption

In this section the total energy consumed by the linear array to really a packet of bits from assigned node to the sink, The way to address high power consumption is the minimization of the peak power required to feed a completely utilized system. In contrast, the energy consumption is defined by the average power consumption over a period of time. Therefore, the actual energy consumption by a data center does not affect the cost of the infrastructure. However, it is reflected in the cost of energy consumed by the system, which is the main component of data operating costs.

According to the transmitting cost in the networks, it can also be shown that the optimal number of hops is the total energy consumed but relay packet of m bits node n to the sink is then

$$P_c = N_T [P_{te}(T_{on} + T_{st}) + P_o(T_{on})] + N_r [P_{re}(R_{on} + R_{st})]$$

Where

$P_{te}$  is power consumed by the transmitter

$P_{re}$  is power consumed by receiver

$P_o$  is output power of transmitter

$T_{on}$  is transmitter "on" time

$R_{on}$  is receiver "on" time

$T_{st}$  is start-up time for transmitter

$R_{st}$  is start-up time for receiver

$N_t$  is the number of times transmitter per unit of time

$N_r$  is the number of times receiver per unit of time

Having the equation the energy consumption measured by in the number or state changes of sensor node with the energy they consumed during the process of transmission. The evaluation is made between power efficiency of observer sensor node while observing start to end time of transmission on a give time period and the same is true for the server power consumption evaluated by  $P_{te}$  power consumed wile transferring data having start to end time in duration of time.

Accommodating multiple client request multiples the energy consumption while ether the client for requesting for observe or for the servers notify or acknowledge state changes . so the proposed approach way of reducing multiple transmissions will also reduce the frequency of client and servers power usage in addition to minimizing network load which is the biggest disturbance of power . bits over a single or multiple hope of constrained devices. Assuming relationship for the energy spent per bit at the transmitter and receiver circuitry  $E_T$  and  $E_R$  can minimize power consumption it is desired to have the transceiver in a sleep mode as much as possible however constantly turning on and off the transceiver also consumes energy to bring it to readiness for transmission or reception.

### 5.5.3 Delivery Time

The delivery time measured against by the number of every single reiteration involving of sending the set number of data observation are represented. All conditionally successful observations were received by the border router within the anticipated lifetime .at the measurement made by titration for each transmission versus time mostly received on allocated time. On the experiment their was

no decrease in signal strength and there was no lost connectivity or packet loss increased knowingly. This experiment of CoAP protocol proves that further aggregation of conditional observation performs adequate way of optimizing multiple loads of requests.

## 5.5.4 Performance Evaluation

### 5.5.4.1 Power consumption

While evaluating how the energy performed, we compared number of sensor nodes against the time of packets arrival on each notification times, while computing the energy consumption both considering with transmitting and listening power and compared with the original work.

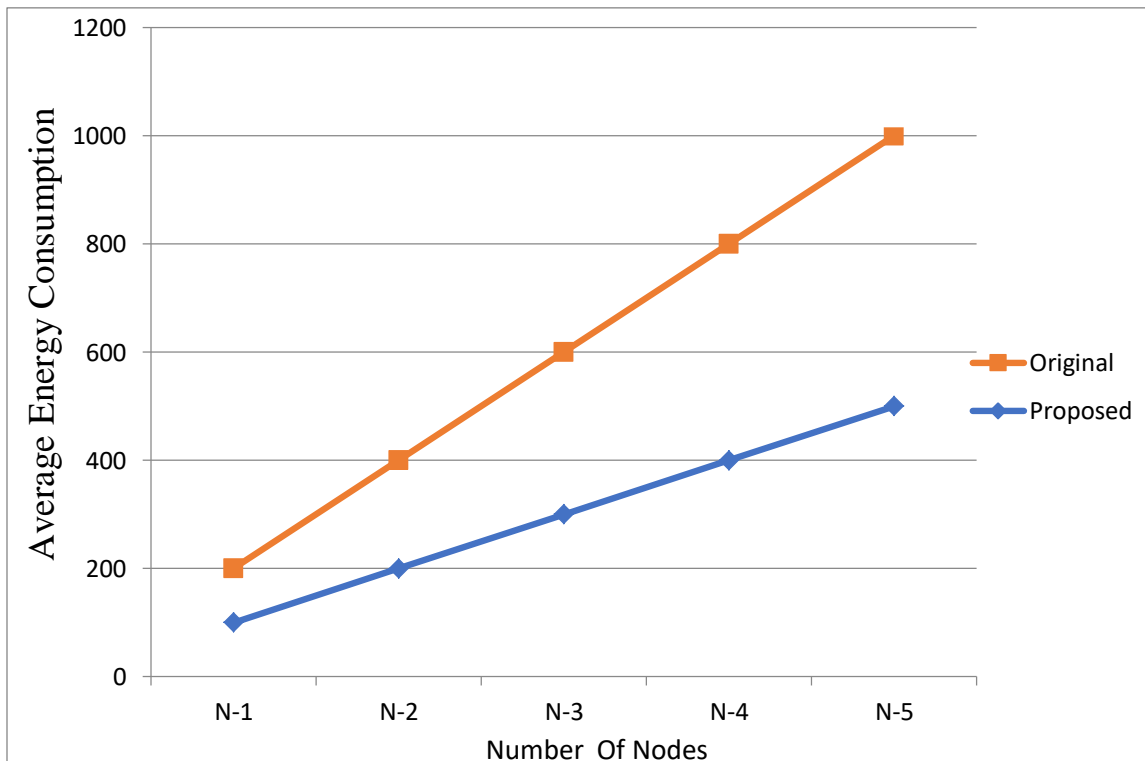


Figure 26: Average energy consumption against number of nodes of the proposed and Original work

### 5.5.4.2 Packet Delivery

During every transaction the packet delivery relationship is obtained by the total number of data packets reached at destination nodes divided by the entire packets which is directed from sources node.

We set nodes to reduce packet delivery delay to have increased link quality sufficient enough to satisfy service quality between nodes and to reduce variations. Frequent variations in the network topology could lead to high path loss between source and destination even if they have closest path.

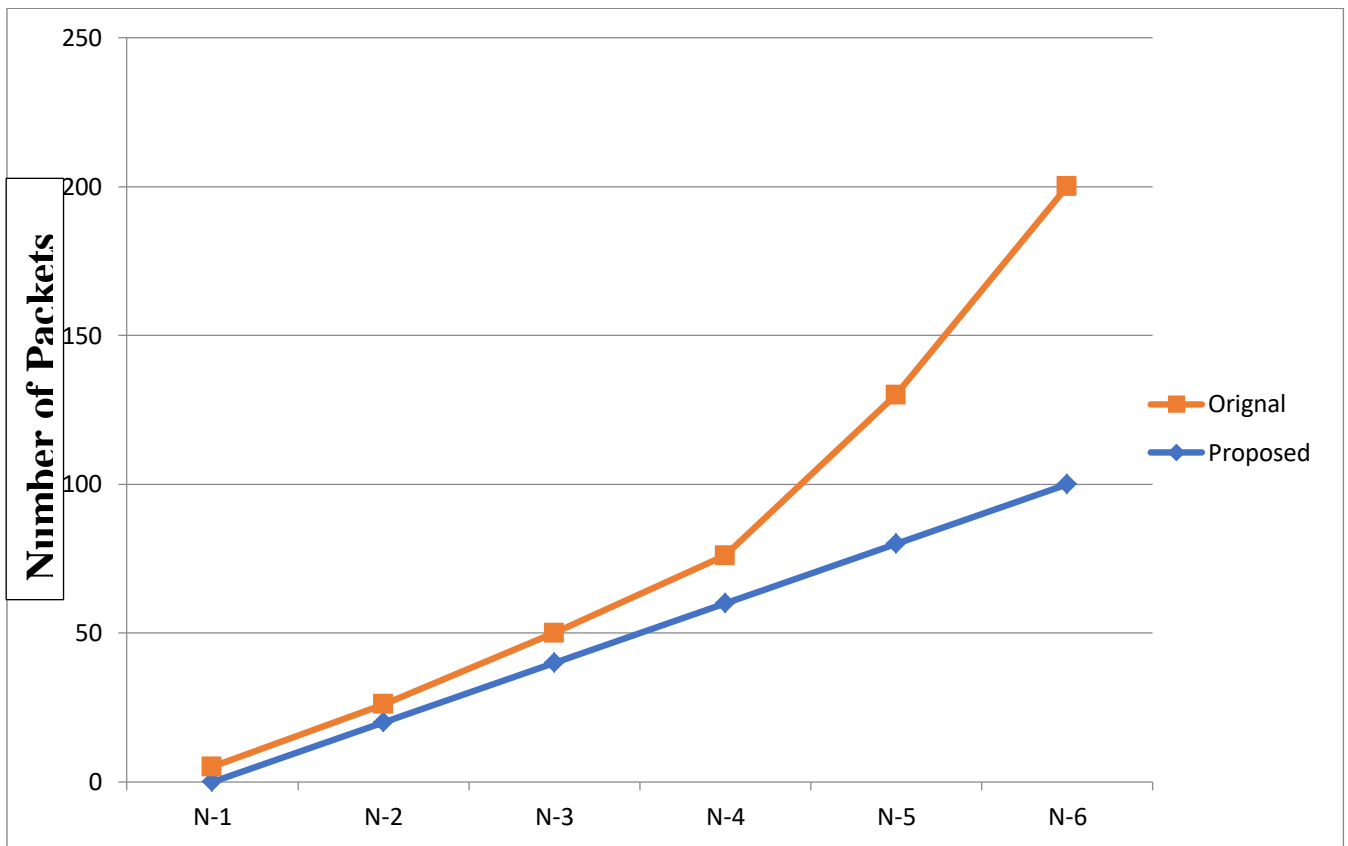


Figure 27: Number of packets transmitted alongside nodes

The average of all packets of original and proposed works are successfully delivered to destination compared to the number of packets that's have been sent out by sender.

### 5.5.4.3 Evaluation Vs Topology

Its important to compare different solution depending on the network topology . as mentioned on the above , improving overall performance is one of our focus which is delay and latency parameter with packets take to get to the destination playing an important role depending on the topology.

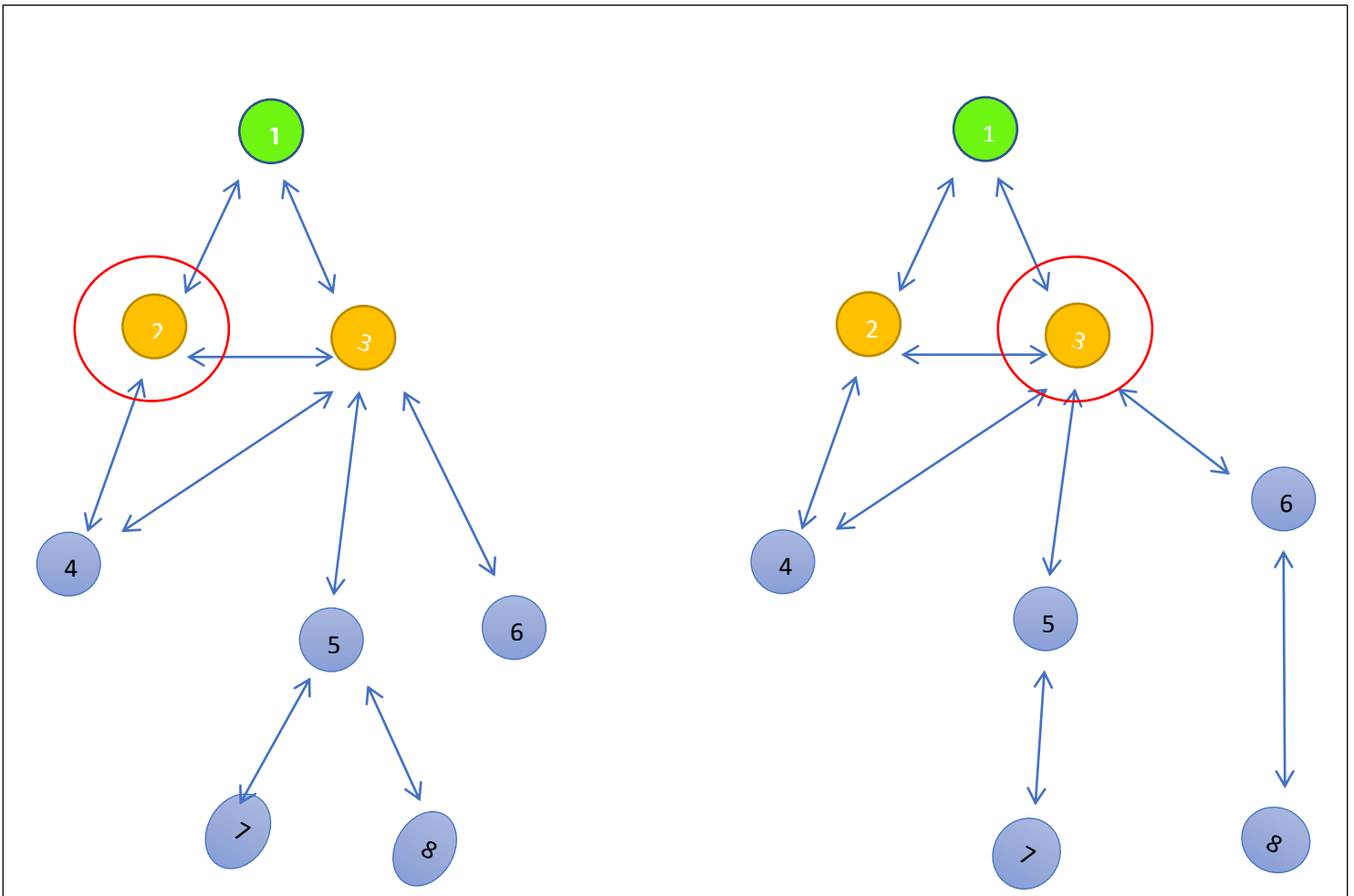


Fig 28: Topologies: Aggregator server on different location

For all condition, latency is computed as the time difference between the occurrence of event at the server and client. For this experiment we just choose both topologies to compare and valuate considering the above.

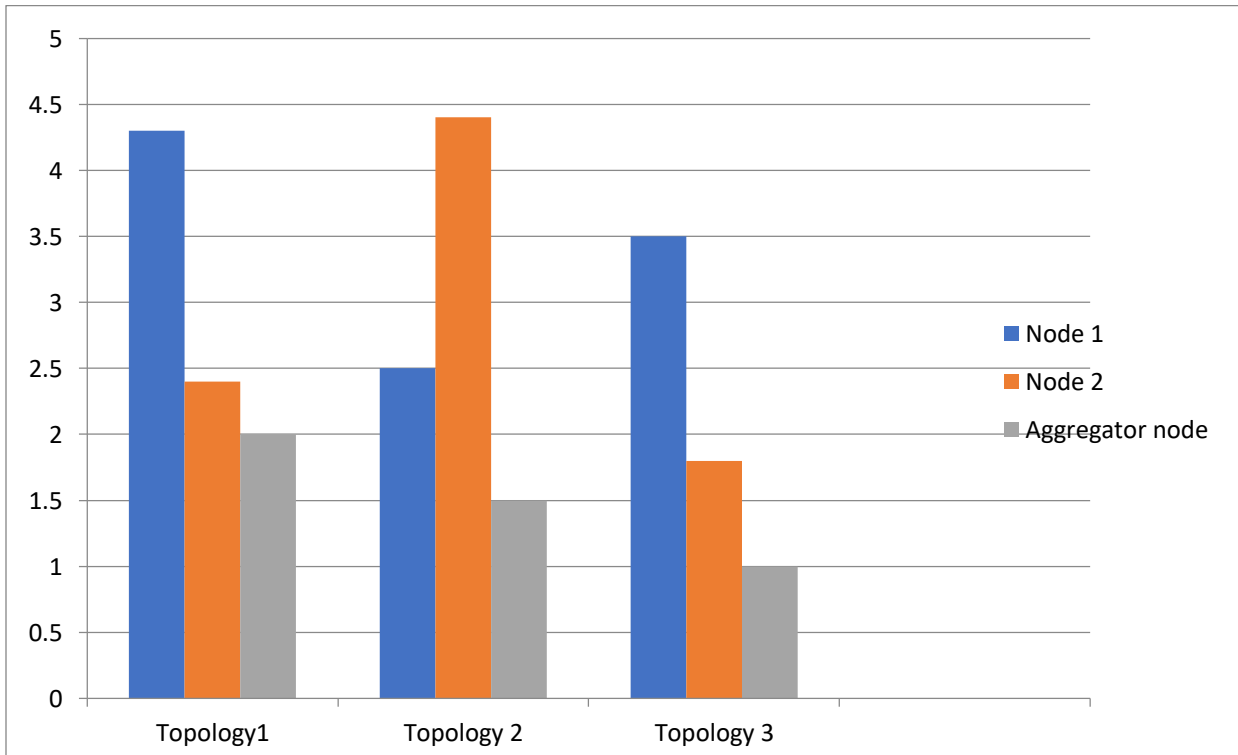


Fig 29: communication delay Vs Topology

#### 5.5.4.4 Notifications loss ratios

As the graph shows , difference in notification rate between experiments. Because of the aggregation of conditional CoAP observation their will only be one observation between nodes. For the experiments, the notification level rises linearly with the number of observers, as the sensor node sends notifications to each client separately. The slope of this linear relation is relational to the notification frequency.

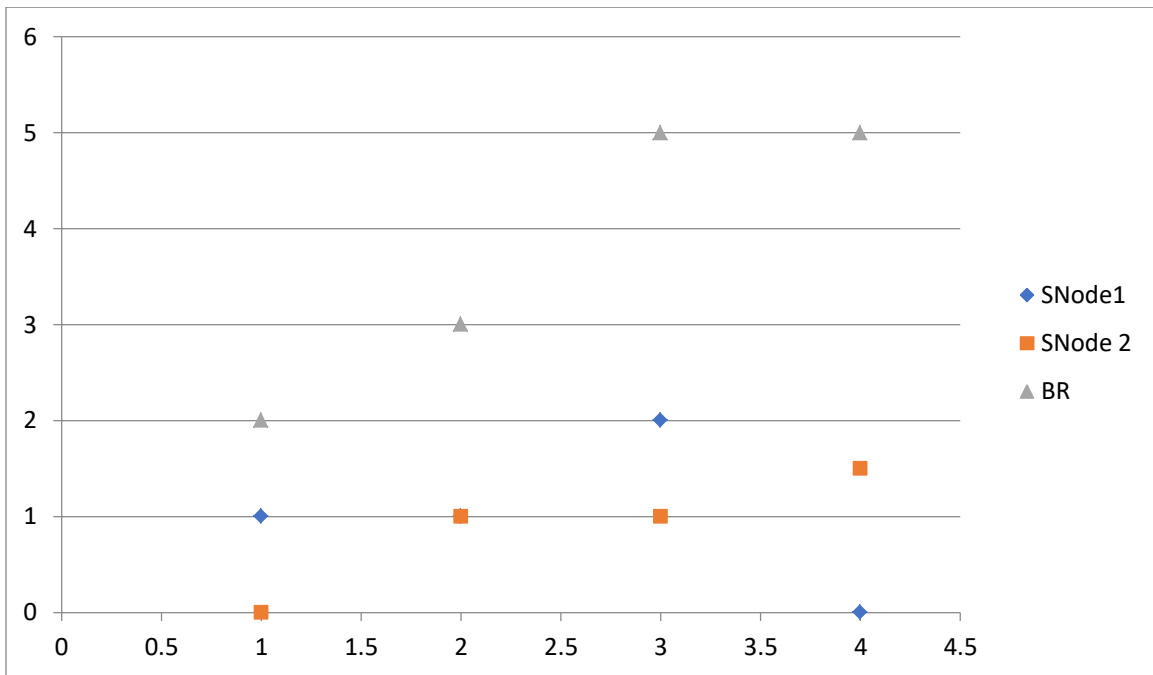


Fig 30: notification packet loss ratios within the number of nodes

All the shown experiment and evaluation help us to show how aggregation help us to overcome limitations addressed on the original paper work by Limited Memory utilization on the sensors nodes with the limited number of simultaneous active multiple CoAP observation relationships. As the message rate reduced by aggregation, helped us to make this experiment is possible

# CHAPTER 6: CONCLUSIONS AND FUTURE WORK

## 6 Future work and conclusion

### 6.1 Conclusion

In the course of this research, we executed several experiments and examined topologies by implementing various aggregation schemes. We evaluated the approach by sending packets from the source node, performing aggregation at the relay and forwarding in the form of an aggregated frame towards the router. This assessment is based on multiple performance parameters such as energy consumption, end-to-end delay, costly transmission, memory usage, overall delay, number of frame transmissions and packet drop. The performed experiments to archive this were capable to demonstrate how packet aggregation found to be performance boost. Our paper work concludes with a brief summary of these tryouts and some possible approaches to provision and more clarified this results. But most of all, having condition option in a well-defined way lets the observers request for different resources conditionally, so this resources evidently notify how they can be retrieved or observed. Those nodes acting like intermediate or head of the cluster nodes can process multiple conditional observations, it makes optimization of observation for multiple requests so much better offering potential to prolong network intelligence, enhance scalability and optimize the lifespan and performance in order to address the desires from the constrained nodes and networks.

### 6.2 Future work

This project widened our acquaintance and improved a lot of experience and competence about packet aggregation in WSN, the implementation of this method in our paper further upgrades the understanding and basic standards on the area, it also allow us be much closer and familiar with IoT and programming language of contiki environment. However, more research and exploration is essential to upgrade and polish quality of this work in near future



### 6.3 Future topics

After going through all the experiments and research of constrained nodes in WSN, we found some approaches which can be an interesting way to explore this topic in more advance manner. First, we examined CoAP protocol with a simple aggregation and experimental way of letting observer clients to observe conditionally, considering only few nodes. However, this topology with proposed approach of optimization can be increased to more nodes and lower power and reduced memory space to observe conditionally the efficiency of packet aggregation on higher scale which may lead towards more precise and improved implementation of these structures or better for that matter.

## Reference

1. Girum Ketema Teklemariam, Jeroen Hoebeke, Facilitating the Creation of IoT Applications Through Conditional Observations in CoAP, EURASIP Journal on Wireless Communications and Networking, 2013.
2. N. Correia, D. Sacramento Dynamic Aggregation and Scheduling in CoAP/Observe Based Wireless Sensor Networks, IEEE Internet of Things Journal,2015.
3. David Sacramento, Data Storage system for Wireless Sensor Networks UNIVERSIDADE DO ALGARVE,2015
4. W.Colitti,K.Steenhaut,N.DeCaro,B.Buta,andV.Dobrota, “REST Enabled Wireless Sensor Networks for Seamless Integration with Web Applications,” in Mobile Ad hoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on, October 2011, pp. 867 –872.
5. Wikipedia, “Wireless sensor node”, [http://en.wikipedia.org/wiki/Sensor\\_node](http://en.wikipedia.org/wiki/Sensor_node), accessed on July 2015.
6. Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP),” Jun. 2014.
7. “Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content,” 2014.
8. R. Fielding. Representational State Transfer (REST). Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine, 2000.
9. K.Hartke, Observing Resources in CoAP , Universitaet Bermen TZI, January 16, 2014
10. IEEE 802.15.4 Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). IEEE Std. 802.15.4, 2006
11. Zechinelli M JL, Bucciol P, Vargas-Solar G. Energy aware data aggregation in wireless sensor networks. In Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on 2011 Feb 28 (pp. 1-5). IEEE.

12. Lindsey S, Raghavendra C, Sivalingam KM. Data gathering algorithms in sensor networks using energy metrics. *Parallel and Distributed Systems, IEEE Transactions on*. 2002 Sep;13(9):924-35.
13. Shelby, Z.; Hartke, K.; Bormann, C. *The Constrained Application Protocol (CoAP)*; RFC 7252; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2014.
14. Li ST, Hoebeke J, Jara AJ: *Conditional Observe in CoAP: draft-li-core-conditional-observe-03*, IETF Trust. : ; 2012.
15. Hartke, K., "Observing Resources in CoAP", [draft-ietf-core-observe-05](#) (work in progress), March 2012.
- [16] Castro, M., Jara, A., and A. Skarmeta, "Architecture for Improving Terrestrial Logistics Based on the Web of Things", *Sensors* 12, no. 5, 6538-6575, 2012, May 2012.
16. Ketema, G., Hoebeke, J., Moerman, I., Demeester, P., Li, Shi Tao., and A. Jara, "Efficiently observing Internet of Things Resources", *The 2012 IEEE International Conference on Cyber, Physical and Social Computing* November 20-23, 2012, Besançon, France, November 2012
17. Yao Y, Gehrke J. The cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record* 2002; 31: 9–18
18. P. Jesus, C. Baquero, and P. Almeida, "A survey of distributed data aggregation algorithms," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 381–404, 2015.
19. Chi, E.: Introducing wearable force sensors in martial arts. *IEEE Pervasive* 4, 47–53 (2005)
20. Alwan, M., Rajendran, P.J., Kell, S., Mack, D., Dalal, S., Wolfe, M., Felder, R.: A smart and passive floor-vibration based fall detector for elderly. In: *Proceedings of the 2nd International Conference on Information and Communication Technologies*, pp. 1003–1007 (2006)
21. Li, S.T. CoRE Working Group. "Conditional Observe in CoAP" Internet Draft; IETF CoRE Working Group, 14 February 2012. Available online: <http://tools.ietf.org/html/draft-li-core-conditional-observe-04> (accessed on 26 February 2012).
22. Ayyaz Mahmood Salekin Imran, *Packet Aggregation in Telos BWSNs: Design, Implementation and Experiments*, Grimstad, May 16, 2016
23. [Z. Shelby13] Z. Shelby, Sensinode, K. Hartke, "Constrained Application Protocol (CoAP)," *draft-ietf-core-coap-18*. [2013-06--28] <http://tools.ietf.org/html/draft-ietf-core-coap-18>
24. S . Thombre, R.U. Islam, K. Andersson, M.S. Hossain Performance analysis of an ip based protocol stack for wsns 2016 *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2016), pp. 360-365, 10.1109/INFOCOMW.2016.7562102 CrossRefView

25. Y. Chen, T. Kunz Performance evaluation of iot protocols under a constrained wireless access network 2016 International Conference on Selected Topics in Mobile Wireless Networking (MoWNeT) (2016), pp. 1-7, 10.1109/MoWNet.2016.7496622
26. M. Collina, M. Bartolucci, A. Vanelli-Coralli, G.E. Corazza Internet of things application layer protocol analysis over error and delay prone links 2014 7th Advanced Satellite Multimedia Systems
27. J.J. Lee, S.M. Chung, B. Lee, K.T. Kim, H.Y. Youn Round trip time based adaptive congestion control with coap for sensor network 2016 International Conference on Distributed Computing in Sensor Systems (DCOSS) (2016), pp. 113-115, 10.1109/DCOSS.2016.35
28. sumeet thombre, performance analysis of IP based WSNs in real time systems Lulea University Of Tecnology ,2016
29. Thombre, S., Ul Islam, R., Andersson, K., and Hossain, M.S. 2016. Performance Analysis of an IP based Protocol Stack for WSNs. in Proceedings of the 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, Piscataway, NJ, pp. 691-696
30. Liu, Y.; Dillon, T.; Yu, W.; Rahayu, W.; Mostafa, F. Missing value imputation for Industrial IoT sensor data with large gaps. IEEE Internet Things J. 2020, 7, 6855–6867.
31. Hua, C.; Yum, T.S.P. Optimal routing and data aggregation for maximizing lifetime of wireless sensor networks. IEEE/ACM Trans. Netw. 2008, 16, 892–903.
32. . Shelby, Z., Hartke, K. and C. Bormann, "[The Constrained Application Protocol \(CoAP\)](#)", RFC 7252, DOI 10.17487/RFC7252, June 2014.
33. 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

# Appendixes

In this chapter, we clarify the methodological configuration by providing a brief user manual for enhanced understanding. In second section, we provide our programming codes for each scheme which are labeled in Chapter 4.

## Appendix A

### User Manual

Before beginning this guide, we presume that reader is already aware with the Linux environment and has basic knowledge about Contiki sensor motes. This manual shows configuration and compiling a contiki application.

1. Before selecting any example, we make sure we are compiling this as a root user following command and password.

*sudo -s*

*Password: user*

2. Choose application module to modify the protocol on *contiki-2.7 apps* directory in *er-coap-07* on both modules of *er-coap-07-observing.h* and *er-coap-07-observing.c*

*/home/user/contiki-2.7/apps/er-coap-07*

3. As a edge of our network ,we choose border router to help us connect to the another end

*/home/user/contiki-2.7/example/ipv6/rpl-border-router/border-router.c*

4. We choose our both servers form contiki examples

*Contiki/examples/er-rest-example/er-example-client.c*

*Contiki/examples/er-rest-example/er-example-server.c*

## Appendix B

In this section, we are providing codes for application module with respect to client / server nodes.

Application module / *er-coap-07-observing.c*

```
#include <stdio.h>
#include <string.h>

#include "er-coap-07-observing.h"

#define DEBUG 0
#if DEBUG
#define PRINTF(...) printf(__VA_ARGS__)
#define PRINT6ADDR(addr)
PRINTF("[%02x%02x:%02x%02x:%02x%02x:%02x%02x:%02x%02x:%02x%02x%02x%02x%02x%02x]",
((uint8_t *)addr)[0], ((uint8_t *)addr)[1], ((uint8_t *)addr)[2],
((uint8_t *)addr)[3], ((uint8_t *)addr)[4], ((uint8_t *)addr)[5], ((uint8_t *)addr)[6],
((uint8_t *)addr)[7], ((uint8_t *)addr)[8], ((uint8_t *)addr)[9],
((uint8_t *)addr)[10], ((uint8_t *)addr)[11], ((uint8_t *)addr)[12], ((uint8_t *)addr)[13],
((uint8_t *)addr)[14], ((uint8_t *)addr)[15])
#define PRINTLLADDR(lladdr) PRINTF("[%02x:%02x:%02x:%02x:%02x:%02x]", (lladdr)->addr[0],
(lladdr)->addr[1], (lladdr)->addr[2], (lladdr)->addr[3], (lladdr)->addr[4], (lladdr)->addr[5])
#else
#define PRINTF(...)
#define PRINT6ADDR(addr)
#define PRINTLLADDR(addr)
#endif

MEMB(observers_memb, coap_observer_t, COAP_MAX_OBSERVERS);
LIST(observers_list);

/*-----*/
-----*/
#ifndef CONDITIONAL_OBS
coap_observer_t *
coap_add_observer(uiplib_ipaddr_t *addr, uint16_t port, const uint8_t *token,
size_t token_len, const char *url)
{
    /* Remove existing observe relationship, if any. */

```

```

void
coap_remove_observer(coap_observer_t *o)
{
    PRINTF("Removing observer for /%s [0x%02X%02X]\n", o->url, o->token[0], o-
>token[1]);

    memb_free(&observers_memb, o);
    list_remove(observers_list, o);
}

int
coap_remove_observer_by_client(uiplib_ipaddr_t *addr, uint16_t port)
{
    int removed = 0;
    coap_observer_t* obs = NULL;

    for (obs = (coap_observer_t*)list_head(observers_list); obs; obs = obs->next)
    {
        PRINTF("Remove check client ");
        PRINT6ADDR(addr);
        PRINTF(":%u\n", port);
        if (uiplib_ipaddr_cmp(&obs->addr, addr) && obs->port==port)
        {
            coap_remove_observer(obs);
            removed++;
        }
    }
    return removed;
}

int
coap_remove_observer_by_token(uiplib_ipaddr_t *addr, uint16_t port, uint8_t
*token, size_t token_len)
{
    int removed = 0;
    coap_observer_t* obs = NULL;

    for (obs = (coap_observer_t*)list_head(observers_list); obs; obs = obs->next)
    {
        PRINTF("Remove check Token 0x%02X%02X\n", token[0], token[1]);
        if (uiplib_ipaddr_cmp(&obs->addr, addr) && obs->port==port && obs-
>token_len==token_len && memcmp(obs->token, token, token_len)==0)
        {
            coap_remove_observer(obs);
            removed++;
        }
    }
    return removed;
}

```

```

int
coap_remove_observer_by_url(uip_ipaddr_t *addr, uint16_t port, const char
*url)
{
    int removed = 0;
    coap_observer_t* obs = NULL;

    for (obs = (coap_observer_t*)list_head(observers_list); obs; obs = obs-
>next)
    {
        PRINTF("Remove check URL %p\n", url);
        if (uip_ipaddr_cmp(&obs->addr, addr) && obs->port==port && (obs-
>url==url || memcmp(obs->url, url, strlen(obs->url))==0)
        {
            coap_remove_observer(obs);
            removed++;
        }
    }
    return removed;
}

int
coap_remove_observer_by_mid(uip_ipaddr_t *addr, uint16_t port, uint16_t
mid)
{
    int removed = 0;
    coap_observer_t* obs = NULL;

    for (obs = (coap_observer_t*)list_head(observers_list); obs; obs = obs-
>next)
    {
        PRINTF("Remove check MID %u\n", mid);
        if (uip_ipaddr_cmp(&obs->addr, addr) && obs->port==port && obs-
>last_mid==mid)
        {
            coap_remove_observer(obs);
            removed++;
        }
    }
    return removed;
}
/*-----*/
void
coap_notify_observers(resource_t *resource, int32_t obs_counter, void
*notification)
{
    coap_packet_t *const coap_res = (coap_packet_t *) notification;
    coap_observer_t* obs = NULL;
    uint8_t preferred_type = coap_res->type;

```



```

/* Iterate over observers. */

for (obs = (coap_observer_t*)list_head(observers_list); obs; obs = obs-
>next)
{
    if (obs->url==resource->url) /* using RESOURCE url pointer as handle
*/
    {
        coap_transaction_t *transaction = NULL;

        /*TODO implement special transaction for CON, sharing the same
buffer to allow for more observers. */

        if ( (transaction = coap_new_transaction(coap_get_mid(), &obs->addr,
obs->port)) )
        {
            PRINTF("          Observer ");
            PRINT6ADDR(&obs->addr);
            PRINTF(":%u\n", obs->port);

            /* Update last MID for RST matching. */
            obs->last_mid = transaction->mid;

            /* Prepare response */
            coap_res->mid = transaction->mid;
            coap_set_header_observe(coap_res, obs_counter);
            coap_set_header_token(coap_res, obs->token, obs->token_len);

            /* Use CON to check whether client is still there/interested after
COAP_OBSERVING_REFRESH_INTERVAL. */
            if (stimer_expired(&obs->refresh_timer))
            {
                PRINTF("          Refreshing with CON\n");
                coap_res->type = COAP_TYPE_CON;
                stimer_restart(&obs->refresh_timer);
            }
            else
            {
                coap_res->type = preferred_type;
            }

            transaction->packet_len = coap_serialize_message(coap_res,
transaction->packet);

            coap_send_transaction(transaction);
        }
    }
}
}
/*-----handling observe request -----*/

```

```

void
coap_observe_handler(resource_t *resource, void *request, void *response)
{
    coap_packet_t *const coap_req = (coap_packet_t *) request;
    coap_packet_t *const coap_res = (coap_packet_t *) response;

    static char content[16];

    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 (coap_add_observer(&UIP_IP_BUF->srcipaddr, UIP_UDP_BUF->srcport,
coap_req->token, coap_req->token_len, resource->url))
            {
                coap_set_header_observe(coap_res, 0);
                /*
                 * For demonstration purposes only. A subscription should return
the same representation as a normal GET.
                 * TODO: Comment the following line for any real application.
                 */
                coap_set_payload(coap_res, content, sprintf(content,
sizeof(content), "Added %u/%u", list_length(observers_list),
COAP_MAX_OBSERVERS));
            }
            else
            {
                coap_res->code = SERVICE_UNAVAILABLE_5_03;
                coap_set_payload(coap_res, "TooManyObservers", 16);
            } /* if (added observer) */
        }
        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) */
    }
}

```

## Appendix C

Application module / *er-coap-07-observing.h*

```
#ifndef COAP_OBSERVING_H_
#define COAP_OBSERVING_H_

#include "sys/stimer.h"
#include "er-coap-13.h"
#include "er-coap-13-transactions.h"

#ifndef COAP_MAX_OBSERVERS
#define COAP_MAX_OBSERVERS COAP_MAX_OPEN_TRANSACTIONS-1
#endif /* COAP_MAX_OBSERVERS */

/* Interval in seconds in which NON notifies are changed to CON
notifies to check client. */
#define COAP_OBSERVING_REFRESH_INTERVAL 60
#if COAP_MAX_OPEN_TRANSACTIONS<COAP_MAX_OBSERVERS
#warning "COAP_MAX_OPEN_TRANSACTIONS smaller than COAP_MAX_OBSERVERS:
cannot handle CON notifications"
#endif
#ifdef CONDITIONAL_OBS
#include<clock.h>
/*Def Condition*/
typedef struct conditional_obs{
    conditional_obs_allvalues_less = 4,
    conditional_obs_allvalues_great = 5,
    conditional_obs_range =2,
    conditional_obs_min_resptime=1,
    conditional_obs_min_resptime=2,
    conditional_obs_val_equal=5,
    conditional_obs_val_not_equal=6,
    conditional_obs_val_equal=1,
    conditional_obs_val_periodic=7
} conditional_obs_t;

/*Values*/
typedef struct {
    conditional_obs_val_int=0,
    conditional_obs_val_fl = 2,
    conditional_obs_val_durat = 1,
} conditional_obs_val_t;
/*build flags*/
typedef struct {
    con = 0,
    non = 1,
}conditional_obs_flags_t;
```

```

/*Store and Display Information Using Structure*/
typedef struct conditional_obs{
    uint8_t conditional_obs;
    uint8_t conditional_obs_flags;
    uint8_t conditional_obs_val;
    uint32_t values;
} conditional_obs_t;

#endif /*conditional_obs*/

#ifdef CONDITIONAL_OBS
    conditional_obs_t conditionals;
    unsigned long set_of_time;
    uint32_t all_cond;
    uint8_t reg_cond;
#endif
} coap_observer_t;

list_t coap_get_observers(void);

#ifdef CONDITIONAL_OBS /*adding conditional observers*/
coap_observer_t *coap_add_observer(uip_ipaddr_t *addr, uint16_t port,
const uint8_t *token, size_t token_len, const char *url);
int conditional_encode_obs (conditional_obs_t *cond, uint8_t
*encoded_cond,uint8_t *conditional_len);

int accomplish_conditional(coap_observer_t *obs, uint32_t cond_val);
int conditional_set_obs(void *packet, uint8_t *conditional,uint8_t
conditional_len);
int conditional_get_obs(void *packet, uint8_t **conditional);
/*****/.....>>>>>>
#else /*Add normal observeer*/
coap_observer_t *coap_add_observer(uip_ipaddr_t *addr, uint16_t port,
const uint8_t *token, size_t token_len, const char *url);
#endif

int being_observed(const char *url);
void coap_remove_observer(coap_observer_t *o);
int coap_remove_observer_by_client(uip_ipaddr_t *addr, uint16_t port);
int coap_remove_observer_by_token(uip_ipaddr_t *addr, uint16_t port,
uint8_t *token, size_t token_len);
int coap_remove_observer_by_url(uip_ipaddr_t *addr, uint16_t port,
const char *url);
void conditionalservers(resource_t *resource, uint16_t obs_counter,
void *notification);

void conditionalandler(resource_t *resource, void *request, void
*response);

#endif /* COAP_OBSERVING_H_ */

```

