Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
Erasmus Mundus Master's Program in Pervasive Computing & Communications for sustainable Development PERCCOM

Master's Thesis in

PERvasive Computing and COMmunications

for sustainable development

Sumeet Thombre

PERFORMANCE ANALYSIS OF IP BASED WSNs IN REAL TIME SYSTEMS

2016

Supervisors: *Associate Professor Karl Andersson* - (Luleå University of Technology)

Examiners: *Professor Eric Rondeau* - (Universite de Lorraine)

Professor Jari Porras - (Lappeenranta University of Technology)

Associate Professor Karl Andersson - (Luleå University of Technology)

This thesis is prepared as part of a European Erasmus Mundus program PERCCOM - Pervasive Computing and COMmunications for sustainable development.

Co-funded by the Erasmus+ Programme of the European Union

This thesis has been accepted by partner institutions of the consortium (cf. UDL-DAJ, n°1524, 2012 PERCCOM agreement).

Successful defense of this thesis is obligatory for graduation with the following national diplomas:

- Master in Complex Systems Engineering (University of Lorraine);
- Master of Science in Technology (Lappeenranta University of Technology);
- Degree of Master of Science (120 credits) - Major: Computer Science and Engineering, Specialization: Pervasive Computing and Communications for sustainable development (Luleå University of Technology).

ABSTRACT

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
PERCCOM Master Program

Sumeet Thombre

Performance Analysis of IP based WSNs in Real Time Systems

Master's Thesis in
PERvasive Computing and COMmunications
for sustainable development

2016

80 pages, 38 figures, 13 tables, and 3 appendices.

Examiners: *Professor Eric Rondeau* - (Universite de Lorraine)

Professor Jari Porras - (Lappeenranta University of Technology)

Associate Professor Karl Andersson - (Luleå University of Technology)

Keywords: Wireless sensor networks, Internet of things, 6LoWPAN, CoAP, Contiki OS, interoperability.

Wireless sensor networks (WSNs) are the key enablers of the internet of things (IoT) paradigm. Traditionally, sensor network research has been to be unlike the internet, motivated by power and device constraints. The IETF 6LoWPAN draft standard changes this, defining how IPv6 packets can be efficiently transmitted over IEEE 802.15.4 radio links. Due to this

6LoWPAN technology, low power, low cost micro- controllers can be connected to the internet forming what is known as the wireless embedded internet. Another IETF recommendation, CoAP allows these devices to communicate interactively over the internet. The integration of such tiny, ubiquitous electronic devices to the internet enables interesting real-time applications. This thesis work attempts to evaluate the performance of a stack consisting of CoAP and 6LoWPAN over the IEEE 802.15.4 radio link using the Contiki OS and Cooja simulator, along with the CoAP framework Californium (Cf). Ultimately, the implementation of this stack on real hardware is carried out using a raspberry pi as a border router with T-mote sky sensors as slip radios and CoAP servers relaying temperature and humidity data. The reliability of the stack was also demonstrated during scalability analysis conducted on the physical deployment. The interoperability is ensured by connecting the WSN to the global internet using different hardware platforms supported by Contiki and without the use of specialized gateways commonly found in non IP based networks. This work therefore developed and demonstrated a heterogeneous wireless sensor network stack, which is IP based and conducted performance analysis of the stack, both in terms of simulations and real hardware.

Accepted publications as part of the thesis -

1) Thombre, S, UI Islam, R, Andersson, K and Hossain, MS 2016, 'IP based Wireless Sensor Networks: Performance Analysis using Simulations and Experiments', Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JOWUA), Vol. 7, No. 3 (to appear in September 2016).
2) Thombre, S, Ul Islam, R, Andersson, K and Hossain, MS 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.
3) Thombre, S, 2015, 'Belief rule based DSS - flood assessment using WSNs', Geomundus, Lisbon, 2015.

ACKNOWLEDGEMENT -

This thesis was undertaken at the Pervasive and Mobile Computing Research Group, Department of Computer Science, Electrical and Space Engineering, Lulea University of Technology, campus Skellefteå under the guidance of Professor Karl Andersson. This research was financially supported by the Erasmus Mundus Program PERCCOM and the European Commission, and the Swedish Research Council.

I would like to convey my gratitude to Professor Karl Andersson, for his continual support and motivation throughout this thesis work. I would also like to thank Raihan Ul Islam, a PhD student, whom I learned a lot from, during the tenure of this work. Along with Professor Karl and Raihan, I engaged in stimulating discussions about the work which has been of paramount importance. Their skill and expertise in the domain, and about research in general has helped me open new vistas of learning. Special thanks to Professor Eric Rondeau, Shahadat Hossein, Josef Hallberg, Jari Porras and other associated faculty members with this program.

The whole process has been thoroughly enjoyable and even though there were moments of sheer gut wrenching despair when something went wrong (sensors are sadistic sometimes), the moments of elation and triumph far outnumbered them. I also presented some part of this work in San Francisco in April in an IEEE conference and I truly cherish that experience, enabled by this work.

On the personal front, I would like to thank my mother, for her unwavering support and belief in whatever I undertake and especially my grandmother, whose fond memories remain with me. My friends, both back home and the ones whom I met during the course of this program, have been an amazing presence during this endeavor.

Skellefteå, 25th May, 2016

Sumeet Thombre

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

6LBR – 6LoWPAN border router

6LoWPAN – Ipv6 over Low Power Wireless Personal Area Networks

ACK – Acknowledgement

AES-CCM – Advanced Encryption Scheme – Counter with CBC-MAC

BRB – Belief rule base

CBC-MAC - Cipher block Chaining message Authentication code

CAP – Contention Access Period

CFP – Contention Free Period

CCA - Clear Channel Assessment

CoAP - Constrained Application Protocol

CON – Confirmable

CSMA/CA – Carrier Sense Multiple Access/ Collision Avoidance

DAG – Directed Acyclic Graph

DODAG – Destination oriented DAG

ED – Energy Detection

FFD – Full function device

GTS – Guaranteed Time Slot

HTML – Hypertext Markup language

HTTP - Hypertext Transfer Protocol

ICMPv6 – Internet Control Message Protocol version 6

ICT - Information and Communications Technology

IEEE - Institute of Electrical and Electronics Engineers

IETF – Internet Engineering Task Force

IoT - Internet of Things

IP - Internet Protocol

IPSO – IP for smart objects

IPv4 - Internet Protocol version 4

IPv6 - Internet Protocol version 6

LoWPAN – Low Power wireless personal area network

LR-WPAN – Low Rate wireless personal area network

LLN – Low power, Lossy networks

LQI – Link quality Indication

M2M – Machine to machine

MAC - Media Access Control

MID – Message Identifier

MQTT - Message Queue Telemetry Transport

MTU – Maximum Transmission Unit

NAT – Network Address Translators

ND – Neighbor Discovery

OMA-LWM2M – Open mobile alliance – Light weight machine to machine

OS – Operating system

PHY – Physical layer

QoS - Quality of Service

RAM – Random-access memory

RDC – Radio duty cycling

REST – Representational state transfer

RFD – Reduced functional device

RFID – Radio-frequency Identification

RPL – Ripple Routing Protocol

RTO – Retransmission Timeout

RTT - Round-trip Time

SMS – Short messaging service

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

URI - Uniform resource identifier

URL - Uniform resource locator

USB - Universal Serial Bus

Wi-Fi – Wireless Fidelity

WLAN - Wireless Local Area Network

WoT – Web of Things

WSN - Wireless Sensor Network

XML – Extensible Markup language

1. INTRODUCTION

This chapter begins with a foreword to define the term IoT, followed by describing the WSNs that enable IoT applications. It also presents the current state of affairs in wireless sensor networks. It also briefly mentions the motivation and the sustainability aspects of the work. It highlights the research challenges faced by the author and also establishes the objective of the research work. It proceeds with the contributions made by the research and mentions the limitations of this work. The chapter concludes with an outline of the research.

1.1 Foreword

The term IoT has no specific definition. In its special report on the internet of things issues in 2014, IEEE described the phrase the internet of things (IoT) as,

"A network of items, each embedded with sensors – which are connected to the internet" [1].

One project that directly relates to IoT is the IEEE P2413 [2]. The scope of IEEE P2413 is to define an architectural framework, addressing descriptions of various IoT domains, definitions of IoT domain abstractions, and identification of commonalities between different IoT domains. The IEEE P2413 working group defines an IoT architecture framework covering the architectural needs of various IoT application domains. They define the IoT architecture in a three tiered structure.

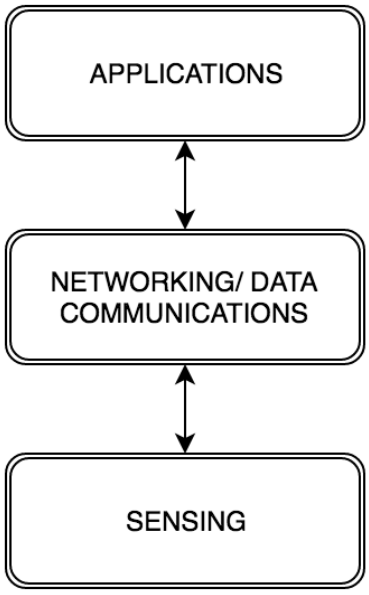

Figure 1 - Three tier architecture of IoT (IEEE P2413)

The Internet Engineering Task Force (IETF) is a large, open, international community of network designers, operators, vendors and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. IETF provides its own description of IoT,

"The basic idea is that IoT will connect objects around us (electronic, electrical, non-electrical) to provide seamless communication and contextual services provided by them. Development of RFID tags, sensors, actuators, mobile phones make it possible to materialize IoT which interact and co-operate with each other to make the service better and accessible anytime, from anywhere [3]."

All these definitions talk about the networking and data communications along with network services, which are realized by wireless sensor networks. These WSNs consist of autonomous sensors to monitor the physical world and its characteristics, such as temperature, humidity etc. and to collaboratively relay their data through the network. In other words, a WSN is a network made up of numerous sensor nodes with sensing and computational capabilities, along with wireless communications. These distributed sensing and communication capabilities along with the simplicity of implementation provided by a wireless communication paradigm make WSNs suitable for real time applications. By providing distributed, real-time information from the physical world, WSNs extend the reach of current cyber infrastructures to the physical world [4].

These WSNs must be designed with sophisticated and extremely efficient communication protocols along with sensors with advanced sensing capabilities. This research attempts to do exactly that, studying the various protocols available today, selecting the most optimal ones, benchmarking them using simulations and then demonstrating them on real hardware. The research work evaluates the different protocols at various layers of the stack on metrics such as interoperability, scalability, reliability, energy efficiency, simplicity etc. to come up with a proposition of a network stack, which will then later be simulated and deployed to conduct basic performance analysis.

The current state of affairs in IoT seems to be present a wide assortment of tools, technologies, protocols, hardware platforms, OS, simulators etc. They are essentially different ways of doing similar things.

Figure 2 – Various popular tools, technologies, OS, hardware platforms, simulators and protocols in the IoT domain today

With so many tools and technologies available, the first step of this research work was to examine the smorgasbord of possibilities. Industry standards, white papers, academic papers and journals were used for literature survey to come up with a stack which would be then implemented. At the network layer, undeniably, ZigBee comes out as the popular choice for low-cost, low-power wireless standard among the ones available currently. The ZigBee alliance has a lot of industrial partners, who have adopted this standard [5]. An IETF recommendation, 6LoWPAN [6] entered the fray as a competition to ZigBee. The biggest advantage in its favor, is that it allows for seamless integration with other IP based systems. This concept of interoperability is an important one, and is the most important metric in selecting an optimum stack in this work. On the application side of things too, another IETF recommendation, CoAP [7] is making waves since its introduction. CoAP is optimized to be small, light and fast for M2M/IoT applications. Like HTTP, it has very little logic/semantics embedded in it, and instead uses a limited command set to communicate with RESTful server processes. Unlike HTTP, it's designed to work on the UDP (or a UDP-like) protocol, instead of TCP/IP. This work uses CoAP as the application layer protocol, as it is RESTful, lightweight and facilitates interactivity based on resource discovery and support for content negotiation. This makes it a formidable choice along with 6LoWPAN to form what is known as the wireless embedded internet.

This research is important as it helps to unify and synergize efforts in the wireless sensor network industry. Establishing an open, standardized network stack with protocols catering to the needs of the constrained devices in terms of limited memory, low power and processing capability is quintessential for the development of smart city applications. This work proposes such a network stack and simulates it to obtain basic performance parameters in varying network conditions. Eventually, the simulated work is also demonstrated on real hardware and scalability analysis is conducted. The proliferation of such WSN technologies is enabling industries and academia to come up with several interesting real time systems including disaster recovery management systems like flood detection, which is the pilot case for this study.

This research work falls directly under the ambit of ICT for sustainability. The proposition of the stack made helps make possible smart city/ IoT applications like flood detection. Also, using IP based stacks helps solve the problem of interoperability, which is the main issue this research tries to tackle. Non IP based communications use application gateways to connect to the global internet, thus requiring additional hardware. Moreover, vendors manufacture their products to be used with their own software built on top, making them incompatible with other devices in the same radio frequency range. Open, standardized protocols, tools, hardware platforms, OSes, simulators etc. are the urgent need for IoT today, and this research work implements a flood detection application based on these. The use case selected of flood detection is designed with sustainability in its core ethos, it benefits all three pillars of people, planet and profit. So there is a two-level sustainability impact of this work,

1. Usage of an IP based stack to foster interoperability and therefore reduce additional hardware and promote open standards over proprietary ones. This also prevents vendor lock-in, encourages reusability and sharing of resources thus reducing efforts, time and money.
2. The application of flood detection itself, where this stack is used, to serve as a disaster recovery management system causing invaluable savings to property, planet and most importantly, human resources. The system is very cheap to deploy as demonstrated in this work and costs about 650 Euros (small scale deployment) and since the sensors are battery powered, it lasts for years.

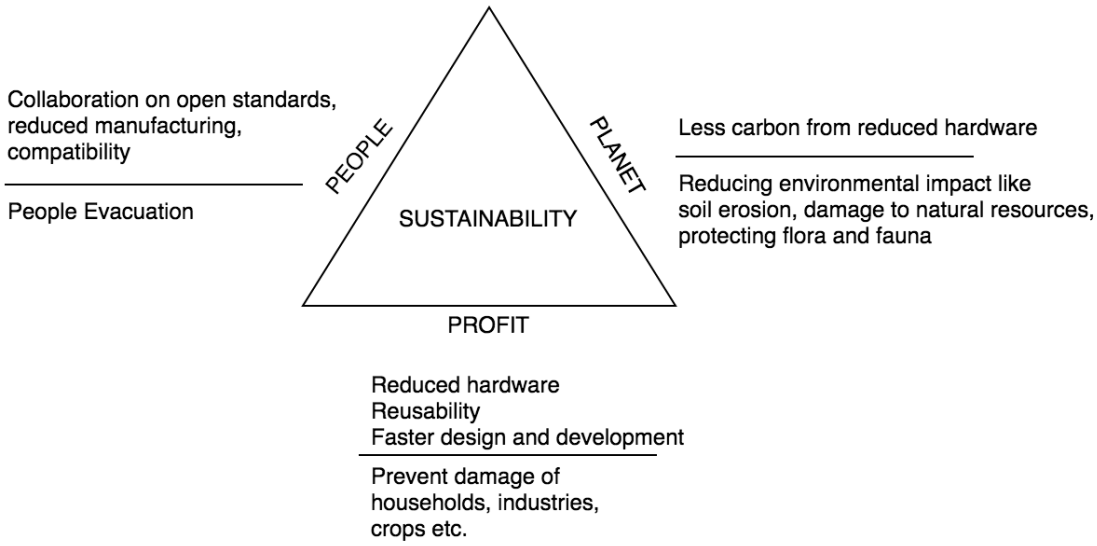

Figure 3 – The sustainability aspects of this work

The work uses an IP based network stack and directly enables the development of smart city applications, like flood detection. It promotes open standards and technologies over proprietary standards and hence fosters interoperability overcoming integration problems prevalent in the industry today. The simulations help understand the performance of the stack in varying network conditions and provide benchmarks. The physical deployment then serves as a validation of the simulated stack. These benchmarks allow better deployment for various smart city applications under varying network conditions.

1.2 Research Challenges

The current state of IoT applications consist of a plethora of proprietary technologies and tools causing integration problems with the Internet and Internet-based services, thus stifling the true potential of the internet of things. This problem of interoperability is an important obstacle this research tries to address. Other important problems to be addressed are that of reliability, simplicity, scalability and ease of deployment. The usage of IP based stacks present challenges to tiny, constrained sensor nodes in terms of memory and processing power. Next come the challenges associated with the performance of the WSNs in variable network conditions such as traffic, noise, topology etc. Ultimately, the cost, reliability and complexity of the physical system provide additional deployment challenges. This research work attempts to address these challenges by using an IP based network stack and conducting proof of concept deployment and performance evaluation of the stack.

1.3 Research Objective

1. The objective of this research is to develop and demonstrate a heterogeneous wireless sensor network architecture suitable for IoT/smart city applications.
2. As proof of concept, a flood detection application is chosen as a pilot, with sensors relaying environmental information such as temperature, humidity, moisture, light etc., using different technologies thereby demonstrating a heterogeneous WSN stack.
3. Another important objective of this research is to conduct performance analysis of the proposed wireless sensor network stack, both in terms of simulations and physical deployments under varying network conditions and compare the simulated and experimental results.

1.4 Research Contribution

This research work studied and reviewed the current state of IoT/smart city technologies and made a proposition of system architecture specifically for a flood detection application, which is chosen as the pilot test case. This proposition was followed by the study of various tools, technologies and network simulators to simulate the proposed architecture and carry out performance analysis in terms of throughput, delay and packet loss. This was followed by the study and review of different hardware platforms and motes, to come up with different scenarios of deployment. Ultimately, a deployment using T-mote sky sensor motes and a raspberry Pi was made, which was then benchmarked for performance and, scalability analysis was carries out. The work earnestly hopes to contribute in a novel way, by providing basic simulation and real deployment performance analysis, which could help potential researchers to better choose platforms and tools, and plan their application's deployment accordingly. The work uses open source tools, operating systems, hardware platforms and technologies, and the code of this work is made publically available and accessible.

1.5 Research Limitations & Scope

The research work itself is not focused with the security aspects of WSNs. Security threats and ways to overcome them using appropriate mechanisms is crucial for IoT/smart city applications, but does not fall into the purview of this work. Experimentation with different MAC/RDC protocols to enhance energy efficiency, although an important concern for

sustainability, also is beyond the scope of this work and is the basis for future work. The flood detection application uses a belief rule based inference methodology for assessments, but its comparison with other similar methodologies is also outside the ambit of this work. Owing to the limited time and resources, detailed comparison and benchmarking between different IoT stacks such as ZigBee, Z-wave etc. with the proposed 6LoWPAN stack, is also not possible. The biggest limitation of this work was the 5-hop scenario in physical deployments, which was not achieved as a topology beyond 3 hops was not achieved by the experimental setup.

1.6 Thesis Outline

The thesis structure is arranged as described below,

Chapter 2 - Background and Literature review

This is followed by the description of the current state of IoT protocols and standards at different layers of the stack. Then, the motivation and advantages of having IP based network stacks is discussed followed by the case study of flood detection. Finally, this chapter talks about belief rule based systems, which is a multiple criteria decision support system.

Chapter 3 – Technologies and Tools

This chapter describes the methodology used for literature survey of IoT tools and technologies, network protocols, reference architectures and applications etc. for knowledge formulation. It conveys explanations of the various tools and technologies involved in this work. At first, the architecture of the system proposed for flood detection is expounded and the different protocols in the architectural stack are explained. Finally, this chapter talks about the operating system for the internet of things Contiki and a CoAP client Californium (Cf), which is used for this work, along with the Copper CoAP client in Firefox.

Chapter 4 – Simulations

This chapter is dedicated to the simulations conducted for the proposed stack. It starts off with the rationale behind conducting simulations and the different simulators surveyed for this

work. Then the simulator tool used for this work, Cooja is introduced, followed by the simulation setup and the consequent results and observations.

Chapter 5 – Physical deployment

This chapter speaks about the various hardware platforms and deployment scenarios thought of in this work. Then, a Contiki based open source solution 6LBR, which is used for this work is explained. This is followed by the experimental setup and deliberation on the results obtained and observations made.

Chapter 6 – Conclusions and Future work

This chapter talks about the conclusions drawn from the simulations and the deployment carried out. Finally, the future directions of this work are hypothesized.

2. BACKGROUND AND LITERATURE REVIEW

2.1 The IoT protocol wars

Many systems today operate in a fairly isolated manner. Enterprise systems are targeted toward business administration and resource planning, while IT solutions involving embedded systems controlling and interacting with physical processes tend to operate in separate stovepipes. The current state of IoT presents a smorgasbord of protocols and technologies. The biggest battle seems to be fought on the network layer of the stack.

IEEE 802.15.4 is an industry standard specifying the PHY and MAC layers for low-rate wireless personal area networks (LR-WPANs). It has over the years become the de-facto standard for wireless sensor networks and is maintained by the IEEE 802.15 working group. Bluetooth low energy and low power Wi-Fi are entering the fray, but are at very nascent stages of their development.

ZigBee is often confused with 802.15.4. It is important to draw a distinction between the two. ZigBee builds on the IEEE 802.15.4 radio link layer and on a proprietary protocol stack, consisting of network, transport and application layer. Z-wave is another proprietary protocol stack offering, specifically designing solutions for smart homes. Recently, 6LoWPAN products are competing with ZigBee in the marketplace, as they build on 802.15.4 – but even better, they can run on other physical layers, and allow for seamless integration with other IP-based systems.

Figure 4 – Big players in the IoT domain today

Interoperability is an important metric to be considered while selecting protocols in the wireless domain. The applications should be agnostic to the PHY link constraints below them. The ZigBee standard defines all layers from its own network layer right up to the application layer above 802.15.4 PHY and MAC layers. Therefore, ZigBee devices can only

communicate with other ZigBee devices. In case of 6LoWPAN, interoperability is ensured as communication is possible with other wireless 802.15.4 devices as well as with devices on any other IP network link (e.g., Ethernet or Wi-Fi) using a border router, which is a simple bridging device explained in chapter 5. An application gateway is essential in case of ZigBee for communicating with non-ZigBee devices to connect it to the global internet.

Also, 6LoWPAN doesn't necessarily append additional headers, which brings down packet overhead and allows more payload to be carried. Also, according to [9] the typical code size for a full-featured stack is 90KB for ZigBee and only 30KB for 6LoWPAN.

When the metric of interoperability became quintessential in the IoT standards today, the ZigBee alliance has recently introduced the ZigBee IP, which uses 6LoWPAN and RPL as its network/routing layers. With this stack redesign, the ZigBee Alliance has acknowledged that defining a standard around a proprietary technology has not worked and that open standards are necessary for interoperability, for delivering long term value and especially for avoiding vendor lock-in.

The Thread group is emerging as a big player in this space, and it uses 6LoWPAN/ RPL too as the networking/ routing layer and 802.15.4 at the physical and MAC layers. It is mainly focused on home automation. The Thread group should make a clear distinction between a thread compliant product and a product which will use 6LoWPAN and IEEE 802.15.4 but which will not have the Thread piece of software built on top. If the IoT sector starts confusing 6LoWPAN with Thread, then the Thread group will have made the same mistake as the ZigBee Alliance with its distinction with 802.15.4. However, it almost seems as though the industry is converging towards uniform standards at the lower layers of the stack and differ only in their application layers!

Coming to the transport layer, UDP remains a light weight protocol option with low overheads and complexity with respect to TCP, and hence is the popular option for WSNs but both MQTT and HTTP use TCP, while CoAP employs UDP. Both CoAP and MQTT are suitable for use at the application layer in the IoT domain. They are open standards, lightweight and suitable to constrained environments than HTTP, allow event based communication, can have IP underneath and have various implementations. MQTT gives flexibility in communication patterns and acts primarily as a channel for data, whereas CoAP is designed for web interoperability.

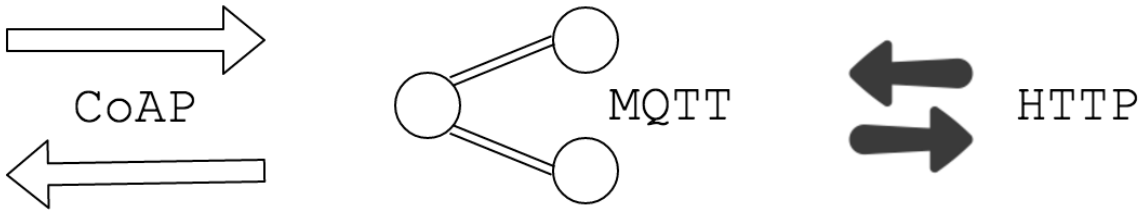

Figure 5 – IoT application layer competing protocols

MQTT and CoAP differ in various aspects. MQTT is a many-to-many communication protocol to relay messages between multiple clients through a central broker. It achieves loose coupling, an important service oriented principle by separating producer and consumer by allowing clients to publish with the onus on the broker to decide where to route messages. CoAP is, primarily, a one-to-one protocol for transferring state information between client and server. It supports the observe method for changes in events which is very sustainable and incorporates congestion control mechanisms over the unreliable UDP at the transport layer.

MQTT clients have a long-lived outgoing TCP connection to a broker, which is already very heavy on constrained devices. CoAP clients and servers both send and receive UDP packets, which is a light weight solution. In NAT environments, tunneling or static port forwarding can be used to allow CoAP. MQTT provides no support for message labelling with types or metadata to help clients understand it. In MQTT, all clients should know the message formats up-front in order to communicate. In CoAP, however, inbuilt support is provided for content negotiation and discovery allowing devices to probe each other to find ways of exchanging data [10]. The application use case dictates the use of one protocol over the other and in this work, CoAP proves to be the right fit for a request response based querying mechanism for sensors.

The fact that major IoT players want to define a standard based on 6LoWPAN, acknowledges the importance of IP and Web technologies for IoT interoperability purpose. However, building pieces of software on top of open standards might anyway hinder interoperability and might not solve the vendor lock-in problem. The Internet has amassed vast success because of interoperability, courtesy of Web technologies RESTful HTTP, without any additional software on top. Using open web technologies such as IETF CoAP might be a better solution than adding pieces of software on top of open standards [11].

2.2 IP based WSNs

When thinking about sensor networks, Internet Protocol (IP) is not the first thing that comes to mind. IP has traditionally been regarded as the protocol for Local Area Networks or Wide Area Networks, PCs and Servers. It was considered unfeasible for Personal Area Networks (PANs) like sensor networks and the sensors themselves because it was considered too heavy for the devices involved in such applications. Recently, however, a large community has started to evaluate many of the misconceptions about IP and have embarked on the development and standardization of 6LoWPAN to inherit the properties that IP has to offer. The IP approach was considered unfeasible to operate on microcontrollers and low-power links, mainly due to the introduced header overhead. However, the IETF 6LoWPAN draft standard changes this, defining how IPv6 packets can be efficiently transmitted over IEEE 802.15.4 radio links. 6LoWPAN defines an adaptation layer to carry IPv6 packets over low data rate, low power, small footprint radio networks (LoWPAN).

Utilizing IP in these WSNs and pushing it to the very edge of the network devices flattens the naming and addressing hierarchy and thereby simplifies the connectivity model [12]. This eliminates the need for complex gateways necessary to translate between proprietary protocols and the standard Internet. We can instead use conventional infrastructure such as switches/routers/bridges, which are well understood, well developed and widely available. Moreover, it promotes reusability by using existing IP tools and technologies.

Also, in the IoT arena, the most common way to communicate with sensors and other wireless low capacity devices has been by the implementation of proprietary protocols over IEEE 802.15.4 radio links. However, proprietary protocols present interoperability problems when trying to transfer packets to devices outside the IEEE 802.15.4 network, and thus, IP connectivity is desired [13].

The biggest advantage of IP based communication in LLNs is to facilitate the use of standard web service infrastructure without using application gateways. Therefore, sensors and other smart objects will be integrated with both, the internet and the web. This integration forms the Web of Things (WoT). The advantage of the WoT is that smart object applications can be built on top Representational State Transfer (REST) architectures [14]. REST architectures exploit the service oriented architecture concept of loose coupling by allowing applications to

share and reuse services. CoAP is based on this REST model and is an embedded web protocol, in which a CoAP resource is an abstraction which is accessible via a coap:// URI.

The advantages of using IP, and thus IoT integration are [15],

- No need to for translation gateways/ additional hardware,
- Optimum usage of current infrastructure,
- Familiarity with IP based technologies, which are proven to work and scale.
- IP is designed to be open, with standard processes and documents available to anyone, encouraging innovation and comprehension,
- Existing tools for managing, configuring, commissioning and troubleshooting IP-based networks.

IP is open, scalable, lightweight, secure, end to end, stable, versatile and is proving to be ubiquitous in communication technologies today. IP has proved to be a formidable invention and is the backbone of the modern internet communications today. Who would have predicted that an invention made four decades ago by Vinton Cerf and Robert Kahn, would be such a force to reckon with? Applying this protocol to WSNs would truly unleash this vision of Internet of Things, where even the smallest of nodes would be connected to the global internet, hence making it an integration protocol.

2.3 Case study – Flood detection

The adverse effects of climate change include the occurrence of extreme weather events. Flooding is one such phenomenon, which causes enormous losses to the planet, people and properties. Since 2000, floods only in Europe have caused at least 700 deaths, the displacement of about half a million people and at least 25 billion EUR in insured economic losses [17]. Around the world today, the consequences of rapid climate change are being experienced as flood risks have increased due to ever rising sea levels. Heavy rains and flash floods have only worsened this problem. It is estimated that 10% of human habitation is in coastal areas, most vulnerable to flooding. Managing flood risks can have several fold advantages,

- Preventable fatalities and injuries,
- Minimize economic losses, properties, crops etc.

- Minimize environmental damages and divert excess water towards productive uses.

Thus, it covers all the three aspects of sustainability – people, profit and planet. Due to these sustainable aspects, flood detection was considered to be the pilot study test case for employing these WSN technologies and this work falls directly in the purview of ICT for sustainability. The need for a decision support system and its stages are explained below.

A decision support system is vital for making flood management decisions including and not limited to, assessment of the flood occurring, limiting the extent of damage caused and awareness among the people. Such a system requires comprehensive support for continuous monitoring of various factors of the environment that characterize flooding, which is achieved by deploying wireless sensors in the environment. The data obtained from these sensors needs to be then stored, processed and analyzed to extract vital data patterns [16].

Flooding is characterized by complex, interrelated and multi-dimensional geophysical processes. Assessment of flooding in stages is a good approach and involves identification, modeling and evaluation. The identification stage is concerned primarily with analyzing contributing factors that cause a flood and the subjects exposed to flooding. The modeling stage involves a maximum likelihood function in estimating a flood occurrence and its consequences to obtain a calculated view of risk. The evaluation stage establishes the aversion limit of flood risks, for further course of actions to tackle the consequences of flooding. All the three pillars of sustainability must be considered, environmental, social and economic, to make it a balanced decision support system [18].

2.4 Belief rule based systems for flood detection

Belief-rule based systems extend traditional IF-THEN logic rule based systems and are capable of capturing complicated non-linear causal relationships between antecedent attributes and consequents. In a BRB system, belief structures are used to represent various types of information and knowledge with uncertainties, and a belief rule is designed with belief degrees embedded in its possible consequents. A Belief Rule Base (BRB) is thus a knowledge representation schema, which allows the capturing of various types of uncertain information. Evidential Reasoning (ER) is used as the inference methodology in the Belief Rule Based Expert System [19].

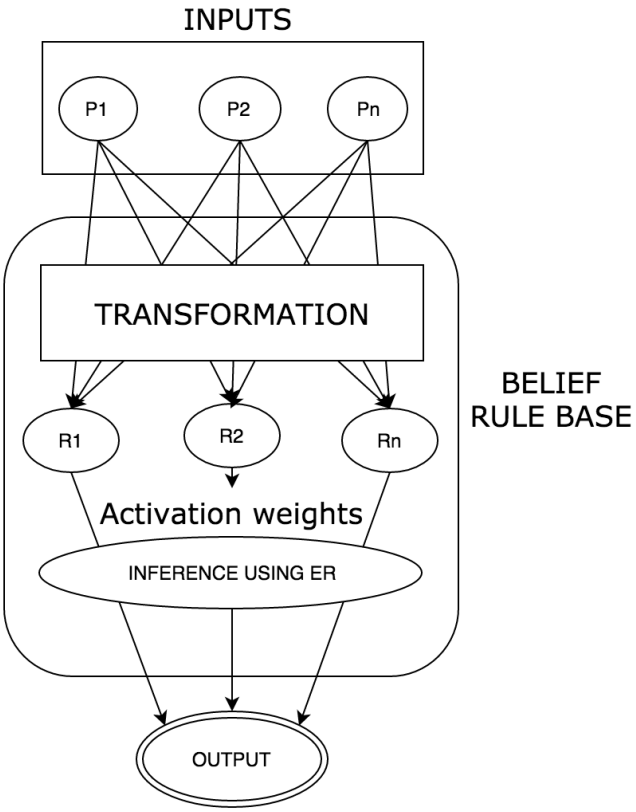

Figure 6 - BRB inference architecture

The flood intensifying factors are identified – meteorological, geophysical, geographical, geomorphological and human activities. The flood dimensions such as area, extent, level and duration are taken into account. A qualitative data set is established and combined with the quantitative data obtained from the sensors deployed, e.g.- rainfall, temperature, soil moisture etc. Both these data sets have attributes which are first transformed and then assigned different weights, inference is carried out using evidential reasoning and an output is generated, on which the decision support system is based. Thus, a belief rule based system combines both quantitative data from the environment and the qualitative expert data to make highly accurate decisions.

3. TECHNOLOGIES AND TOOLS

The methodology and the work flow is briefly described in this chapter. This research work sets out to investigate the available tools, protocols and technologies to best simulate the flood detection application. Thereafter, extensive study is carried out to investigate the hardware platforms available and the best deployment designs are discussed and explained to deploy the system on real hardware in chapter 5. Some of the metrics motivating the choices are interoperability, reliability, simplicity, availability, low power consumption, low memory, low cost etc. This work relies on open standards and proposes a theoretical proposition in the first sub chapter below, which will be expounded with the various protocols involved. Thereafter, the OS and other tools used will be explained.

3.1 Methodology

In order to accomplish the goals mentioned in chapter 1.3, this work will be conducted in three distinct phases:

- a study phase,
- a simulation phase and,
- a physical implementation and evaluation phase.

During the first phase, a comprehensive literature survey about different protocols, tools and technologies of WSNs will be made; then the most common operating systems, network simulators and communication protocols used in WSNs will be analyzed. After these surveys, a solution for the problem statement explained in chapter 1.2 will be designed and then simulated. In the last phase, the physical deployment of the simulated solution will be carried out and performance of the developed solution will be evaluated as a proof of concept for the flood detection use case.

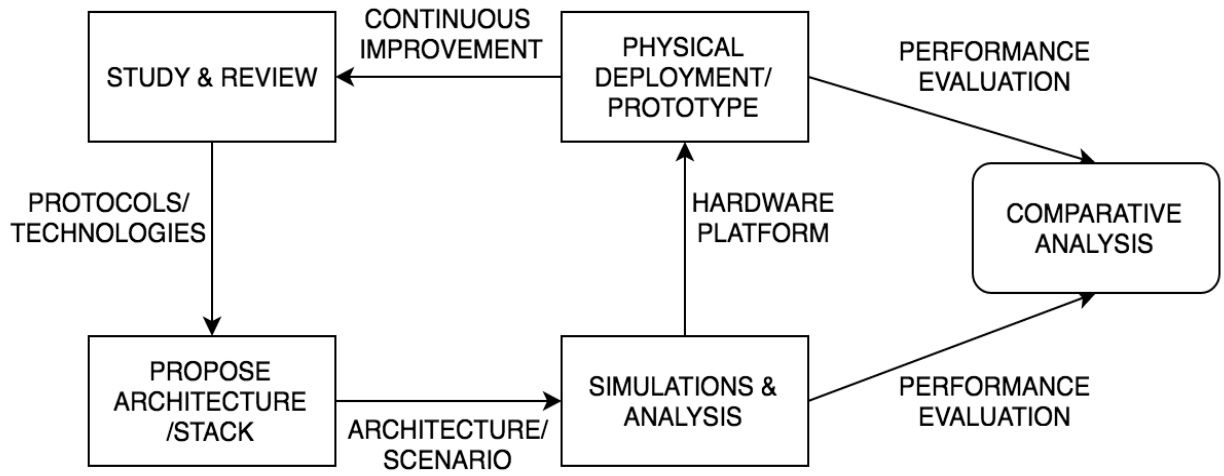

Figure 7 – Methodology and process flow

A review methodology by Kitchenham [8] was employed for the study phase to conduct comprehensive literature survey. It involves three phases, planning, conducting and reporting the review, and is widely used as a guideline for systematic review for software engineering research. It involves the following steps,

- Formulating a review protocol specifying research question being addressed and methods used.
- Developing a search strategy for literature.
- Documenting the search strategy.
- Filtering content using explicit inclusion and exclusion criteria.
- Specifying information obtained from the study including quality criteria to evaluate study.
- Quantitative meta-analysis for summarizing and reviewing previous research work done in the domain.

The review objective was to identify the possible literature about various wireless sensor network architectures that support the IOT and smart city applications. The sources searched to identify primary study material were mainly IETF recommendations, IEEE, Elsevier, ACM databases among others. Industrial white papers and academic books also played a big role in comprehension of concepts and provided a lot of study material. For knowledge formulation, this research included both academic and industrial works. Some search terms and inclusion criteria used were IoT, IoT protocol stack, IoT communication stack, IoT protocols, Smart city protocol stack/ communication, WSN protocols, WSN protocol/communication stack,

Cyber-physical systems etc. After this, the next level search terms were more specific like 6LoWPAN, ZigBee, CoAP, RPL etc. among others.

The study phase, although ongoing continuously, culminated with the theoretical proposition of architectural stack. After this, the simulation phase began, in which different scenarios were thought of, simulators were surveyed, and tools and technologies were explored. This eventually led to the simulations being run using the selected simulator and tools, after which a performance evaluation was carried out. This was followed by the physical deployment phase. Prior to this, a study of various hardware platforms and tools was done in order to come up with different deployment scenarios. Ultimately, a scenario was chosen and deployed on physical hardware and performance evaluation was carried out to compare experimental and simulated results.

3.2 System proposed for flood detection

Andersson, K et. al. [20] proposed a framework for flood detection which takes into account multi-disciplinary characteristics such as meteorological, topographical and geological data, river characteristics, and human activities. There are 17 such input data required to feed the system, which can be both quantitative and qualitative. Examples of quantitative data are rainfall, temperature, humidity, soil infiltration rate, water level etc. and the examples of qualitative data are settlement on flood prone area.

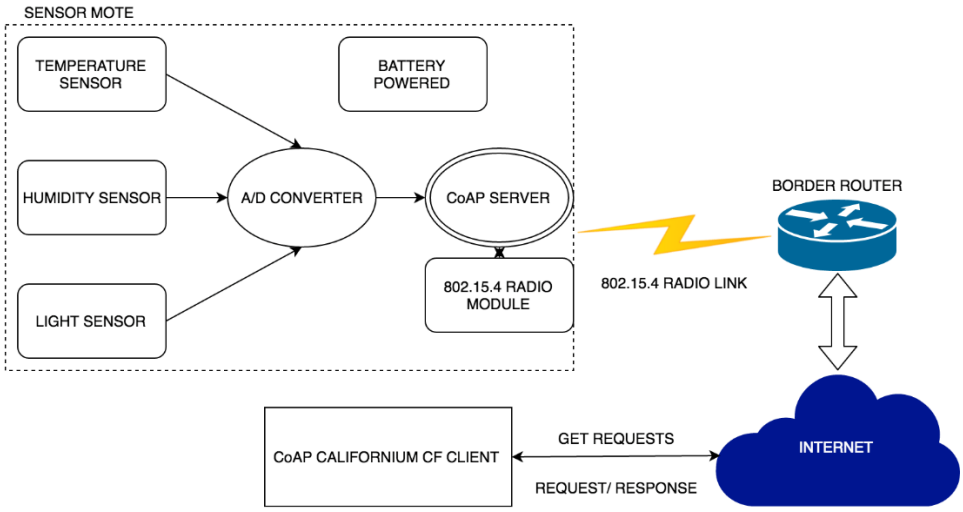

Figure 8 – Proposed network architecture

This work employs sensors like temperature, humidity and light intensity which are exposed over a RESTful CoAP server, and are queried by a CoAP client Californium Cf, which is explained in the next chapter. Chapter 5 speaks about how the border router acts as a gateway to the WSN and connects it to the global internet.

The architectural protocol stack is in accordance with the IETF recommendations. The IPSO Alliance is an open, informal and thought-leading association of like-minded organizations and individuals that promote the value of using the Internet Protocol for the networking of Smart Objects [21]. This IP for smart objects (IPSO) alliance also pushes for the adoption of this stack.

OMA supports the creation of interoperable end-to-end mobile services in the wireless industry. The OMA Lightweight M2M (LWM2M) [22] is particularly suitable for tiny devices with limited processing power, RAM, memory and battery, thus requiring efficient bandwidth usage and recommends the use of this stack.

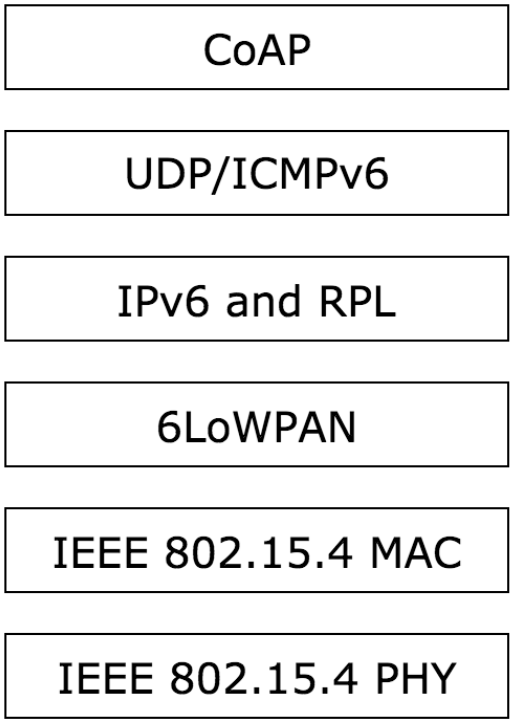

Figure 9 – Protocol stack

3.3 IETF STACK

The architectural stack defined in the earlier chapter mentions various protocols at different layers of the stack which will be explained below,

3.3.1 802.15.4

Wireless personal area networks (WPANs) are used to relay information over relatively small distances. Compared to the local area networks, there is not much infrastructure involved. Therefore, simple, power-efficient, cheap solutions can be implemented for a wide range of devices in WPANs. The 802.15.4 standard defines the physical layer (PHY) and medium access control (MAC) sublayer specifications for low rate, low power wireless connectivity. The highest achievable raw data rate is 250 Kbps but is scalable down to the needs of sensor and automation needs (20 kb/s or below) for wireless communications, making it versatile. The main metrics of these WPANs are ease of installation and troubleshooting, reliable data transfer, low cost, and low power requirements for devices, while maintaining a simple and flexible protocol.

The noteworthy properties provided include:

- Star/mesh/peer to peer topology support,
- Unique 64-bit extended address or allocated 16-bit short address,
- Contention free periods in a super frame structure with optional allocation of guaranteed time slots (GTSs),
- During contention CSMA-CA mechanism,
- Reliable transfers using acknowledgements,
- Very low power,
- Energy detection (ED) and,
- Link quality indication (LQI) [23].

Full-function devices (FFD) and a reduced-function device (RFD) are two categories of devices in 802.15.4 networks. FFDs are capable of serving as PAN coordinators or sensor motes whereas, an RFD is not capable of serving as a PAN coordinator and can only function

as a simple switch or a sensor mote. This work considers the mesh topology as shown in the figure below for the flood detection scenario,

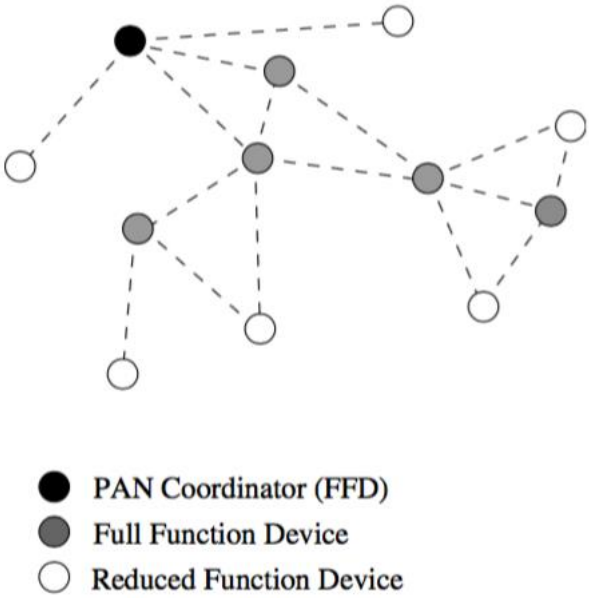

Figure 10 – Mesh topology structure for 802.15.4

The IEEE 802.15.4 architecture is a layered structure as shown in Figure 12. Each layer is responsible for one part of the standard and offers services to the higher layers like in all modular architectures. An LR-WPAN device comprises at least one physical layer, controlling the radio frequency (RF) transceiver along with its low-level drivers and control mechanisms, and a MAC sublayer providing access to the physical channel for data transfer.

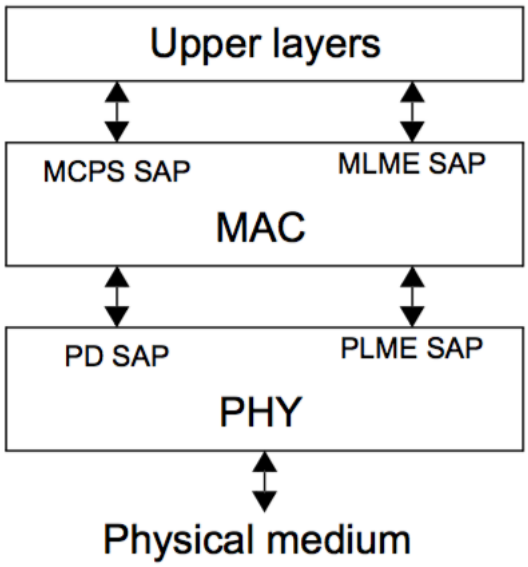

Figure 11 – 802.15.4 layered architecture

The physical layer data service enables the transmission and reception of PHY protocol data units (PPDUs) across the radio channel. Other features of the physical layer include, activation and deactivation of the radio transceiver, clear channel assessment (CCA), channel selection etc. The table below shows the current existing transceivers used in sensor nodes [4].

Table 1 – Current transceivers in WSNs

	Radio				
	RFM TR1000	**Infineon TDA5250**	**TI CC1000**	**TI CC2420**	**Zeevo ZV4002**
Platforms	WeC, Rene Dot, Mica	eyesIFX	Mica2Dot, Mica2 BTnode	MicaZ, TelosB SunSPOT, Imote2	Imote BTnode
Standard	N/A	N/A	N/A	IEEE 802.15.4	Bluetooth
Data rate (kbps)	2.4–115.2	19.2	38.5	250	723.2
Modulation	OOK/ASK	ASK/FSK	FSK	O-QPSK	FHSS–GFSK
Radio frequency (MHz)	916	868	315/433/868/915	2.4 GHz	2.4 GHz
Supply voltage (V)	2.7–3.5	2.1–5.5	2.1–3.6	2.1–3.6	0.85–3.3
TX max (mA/dBm)	12/−1	11.9/9	26.7/10	17.4/0	32/4
TX min (mA/dBm)	N/A	4.9/−22	5.3/−20	8.5/−25	N/A
RX (mA)	1.8–4.5	8.6–9.5	7.4–9.6	18.8	32
Sleep (μA)	5	9	0.2–1	0.02	3.3 mA
Startup time (ms)	12	0.77–1.43	1.5–5	0.3–0.6	N/A

The motes used in the physical deployment use the TI CC2420 chipset shown in the table, which will be explained in chapter 5 more comprehensively. The IEEE 802.15.4 standard also defines a MAC layer, which is based on a super frame structure and relies on the CSMA-CA mechanism. The MAC data service enables the transmission and reception of MAC protocol data units (MPDUs) across the physical layer data service. Other features of the MAC sublayer include beacon management, channel access, GTS management, frame validation, acknowledged frame delivery, association, and disassociation.

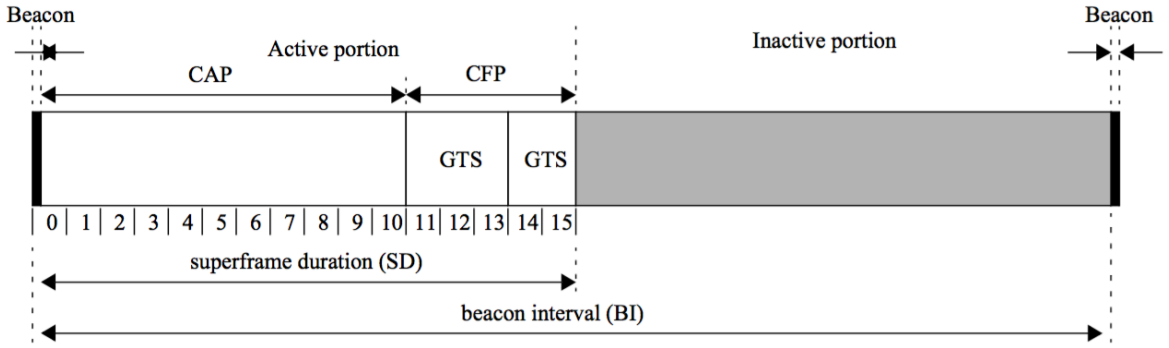

Figure 12 – Super-frame structure defined in 802.15.4

The format of the super frame is defined by the coordinator. The super frame is bounded by network beacons sent by the coordinator, and is divided into 16 slots of equal duration. The beacon interval is the duration between two consecutive beacons and contains the contention access and contention free periods with guaranteed time slots as well as the idle time. During the inactive portion, the coordinator is able to enter a low-power mode. The beacon frame transmission starts at the beginning of the first slot of each super frame. Any device wishing to communicate during the contention access period (CAP) between two beacons competes with other devices using a slotted CSMA-CA mechanism. The super frames also contain a contention free period with optional GTSs, appearing at the end of the active super frame immediately after the CAP.

These characteristics have made it an optimal fit for WSNs where nodes are battery powered and low power, low rate communication is desired. 802.15.4 has thus become the de-facto standard in the industry today at the PHY and MAC layers, and this work also uses this standard for the flood detection scenario.

3.3.2 6LoWPAN/RPL

Various technologies have proliferated on top of the IEEE 802.15.4 standard, in terms of low power networks. Standardization efforts such as 6LoWPAN are focused on providing compatibility between WSNs and existing networks such as the Internet. To integrate WSNs with the Internet, the Internet Engineering Task Force (IETF) is developing the IPv6 over Low-power Wireless Personal Area Network (6LoWPAN) standard [6]. The principle behind this technology is that IP should be applied to even the smallest of devices, thus making sure they are connected to the global internet. IPv6 packets must be carried on 802.15.4 data frames.

The biggest issue and rationale behind having this adaptation layer is the MTU. The MTU size for IPv6 packets over IEEE 802.15.4 is 1280 octets and a full IPv6 packet does not fit in it. Starting from a maximum physical layer packet size of 127 octets and a maximum frame overhead of 25, the resultant maximum frame size at the media access control layer is 102 octets. Link-layer security imposes further overhead, which in the maximum case (21 octets of overhead in the AES-CCM-128 case, versus 9 and 13 for AES-CCM-32 and AES-CCM-64, respectively) leaves only 81 octets available. This is far below the minimum IPv6 packet

size of 1280 octets, a fragmentation and reassembly adaptation layer must be provided at the layer below IP. Furthermore, since the IPv6 header is 40 octets long, this leaves only 41 octets for upper-layer protocols, like UDP. The latter uses 8 octets in the header which leaves only 33 octets for application data [6].

Therefore, the adaptation layer must be provided to comply with the IPv6 requirements of a minimum MTU. However, most applications of IEEE 802.15.4 don't use such large packets. Moreover, small payloads with the proper header compression will produce packets that fit within a single IEEE 802.15.4 frame.

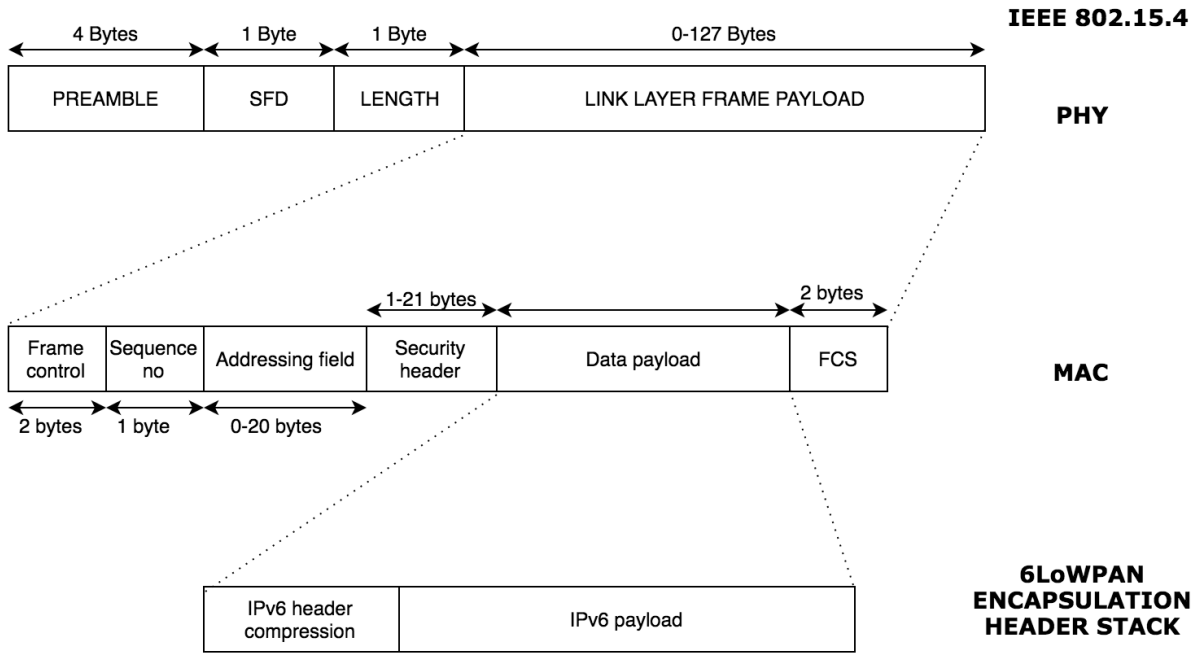

Figure 13 - 6LoWPAN header compression example

Neighbor discovery enables discovery of motes in the vicinity, handling bootstrapping and maintenance issues between IPv6 nodes. The 6LoWPAN working group has defined 6LoWPAN Neighbor Discovery (6LoWPAN-ND) optimized for low-power wireless networks and 6LoWPAN in particular [24]. ICMPv6 [25] enables the maintenance between nodes on IPv6 links.

Other issues that need to be addressed in 6LoWPAN include,

- **Application layer protocols:** HTTP/TCP large payloads of HTML, JSON, XML or SOAP carried over HTTP and TCP, which are of several kilobytes. Such payloads are unsuitable for 6LoWPAN Nodes. This forms the basis for CoAP, an application layer protocol which will be addresses in the next section.
- **Firewalls/ NATs:** When connecting 6LoWPAN through firewalls/NATs there may be problems associated with the blocking of compressed UDP ports, non-standard application protocols used for 6LoWPAN applications, unavailability of static IP addresses etc.
- **IPv4 connectivity:** Today, most of the internet is still in the IPv4 space but 6LoWPAN only supports IPv6 based communication so tunneling mechanisms and address translations should be thought of. The border router has the burden of this interconnectivity.

Handling these issues would ensure full internet integration of WSNs, without the requirement of special gateways and hardware, thus using existing infrastructure like routers, switches etc. to accommodate WSNs on the global internet. Routing in LLNs is explained in the following paragraph.

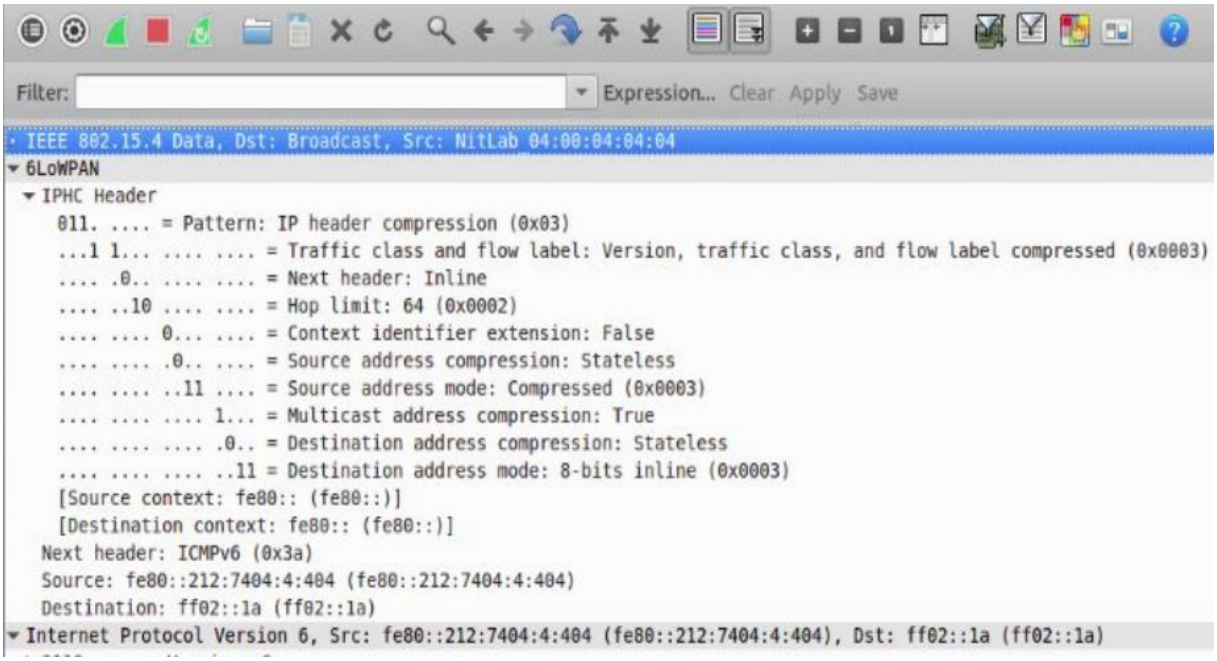

Figure 14 – A Wireshark capture of 6LoWPAN

Routing is the process by which the network determines what path the messages should take through the network. The routing protocol design in LLNs should be sensitive to how much data a network can handle, the speed and the devices' capabilities. Moreover, the nodes are battery powered and therefore poses additional energy efficiency challenges. Therefore, IETF ROLL working group has defined application specific routing requirements for a low-power and lossy networks (LLN) routing protocol RPL. RPL provides a mechanism to disseminate information over the dynamically formed network topology. This dissemination enables minimal configuration in the nodes, allowing nodes to operate mostly autonomously. RPL involves the formation of a directed acyclic graph (DAG) having a DAG root, DODAG (destination oriented DAG) and a DODAG root. The DODAG root may act as a border router for the DODAG; in particular, it may aggregate routes in the DODAG and may redistribute DODAG routes into other routing protocols [26].

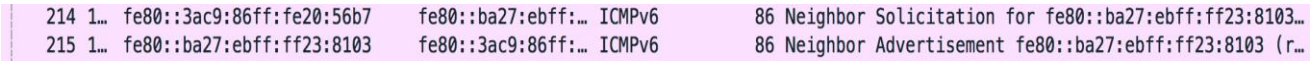

Figure 15 – Wireshark capture showing neighbor advertisement and solicitation messages using ICMPv6

RPL is collection oriented, designed to collect data from source to sink and consists of objective functions which use constraints to compute the best path. The border routers, which normally connect the 6LoWPAN network to other IP networks, typically operate as RPL DODAG roots. RPL proves to be an optimal solution for routing in LLNs in unreliable communication channels for battery powered devices.

3.3.3 CoAP

CoAP is an embedded web transfer protocol suitable for IoT devices. CoAP is RESTful, in which observable resources are made available as server-controlled abstractions and are uniquely identified by coap:// URIs. RESTful methods like GET, PUT, POST and DELETE, like in HTTP are used to manipulate resources [7].

CoAP is not a blind compression of HTTP. HTTP functionalities have been re-designed taking into account the limited processing power and energy constraints of sensor motes. In

addition, various mechanisms have been modified and some new functionalities have been added in order to make the protocol suitable to IoT and M2M applications [14].

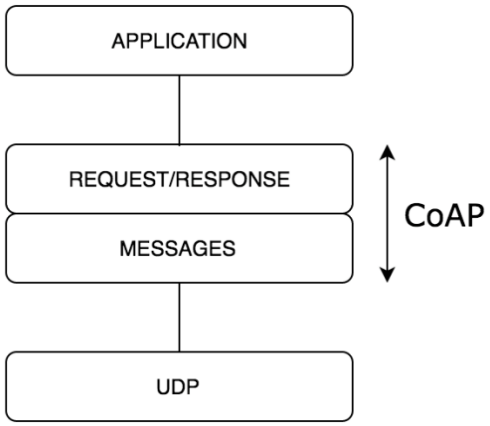

Figure 16 – Abstract layering of CoAP

HTTP and CoAP employ different underlying transport layer. HTTP is built on top of the Transmission Control Protocol (TCP). TCP's flow control mechanisms are heavyweight and not suitable for LLNs and its overhead is considered too high for short-lived transactions, not to forget the power required to keep a TCP connection alive. In addition, TCP does not have multicast support and is rather sensitive to mobility. CoAP uses the User Datagram Protocol (UDP) and therefore has significantly lower overhead and multicast support.

CoAP is split in two layers, transmission and request/response layer. The Transaction layer manages message exchanges between end points. The messages exchanged on this layer can be of four types: CON i.e. Confirmable (reliable mode requiring acknowledgement), NON i.e. Non-confirmable (no acknowledgement), ACK i.e. Acknowledgment (response to a Confirmable request) and RST i.e. Reset (it indicates that a Confirmable message has been receive but context is missing to be processed). The Request/Response layer is where the RESTful communication occurs. This layer is responsible for the transmission of requests and responses for the resource manipulation and transmission. A REST request for e.g. – GET is piggybacked on a Confirmable or Non-confirmable message, while a REST response is piggybacked on the related ACK message with the same message ID.

Reliability is provided by using a message as Confirmable (CON). A Confirmable message is retransmitted using a default timeout (RTO – retransmission timeout) and exponential back-

off between retransmissions, until the recipient sends an Acknowledgement message (ACK) with the same Message ID (in this example, 0x7d34) from the corresponding endpoint, unless the factor CoAP Retransmission factor decrements to 0. This factor is set to 4 by default.

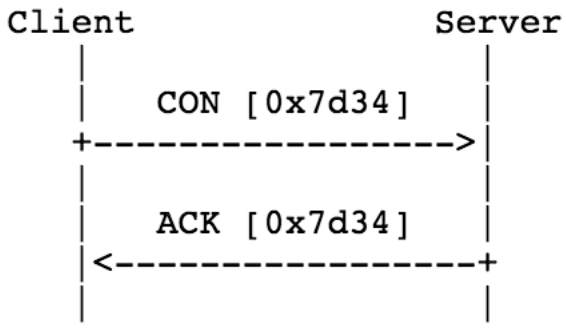

Figure 17 – Reliable message transmission in CoAP CON mode

The retransmission mechanism on the expiration of the RTO timer allows CoAP to have some sort of congestion control over UDP. CoAP uses a short fixed-length compact binary header of 4 bytes followed by compact binary options. A typical GET request during this work was found to be 13 bytes whereas a response was 17 bytes. Since a resource on a CoAP server likely changes over time, the protocol allows a client to constantly observe the resources using the OBSERVE functionality. So the client, who wants to observe registers itself to the resource by means of a modified GET request sent to the server. The server establishes an observation relationship between the client and the resource. Whenever the state of the resource changes, the server notifies each client having an observation relationship with the resource [27]. This is a very energy efficient procedure to keep track of resources based on state change.

[28] studied the power consumption profile of CoAP to compare with HTTP, using Energest, a tool able to estimate the power consumption of the T-mote Sky platform. The results have been taken in steady state conditions and are as shown in the table below,

Table 2 – Energy comparison between CoAP and HTTP

	Bytes per-transaction	Power	Lifetime
CoAP	154	0.744 mW	151 days
HTTP	1451	1.333 mW	84 days

As illustrated in Table 2, an HTTP transaction has ~ 10 times bytes involved than the CoAP transaction, courtesy of the significant header compression in CoAP. CoAP uses a short fixed length compact binary header of 4 bytes and a typical request has a total header of about 10-20 bytes. The higher number of bytes in HTTP reflects in the higher power consumption as a result shown in table 2.

These factors accentuate the viability and motivate the usage of CoAP in our WSN stack for the flood detection scenario. Below are some figures from a running measurement with real sensors. The Californium client is used to send CoAP GET requests (seen as CON – confirmable messages) and the sensors which are running CoAP servers reply (seen as ACK messages) with temperature and humidity data.

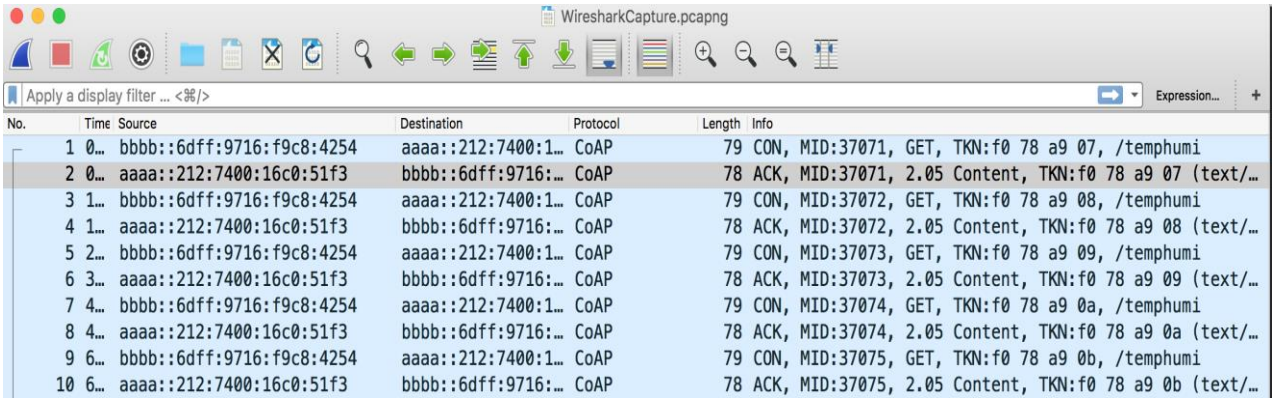

Figure 18 – A live Wireshark capture of CoAP requests (CON mode) and responses (ACK)

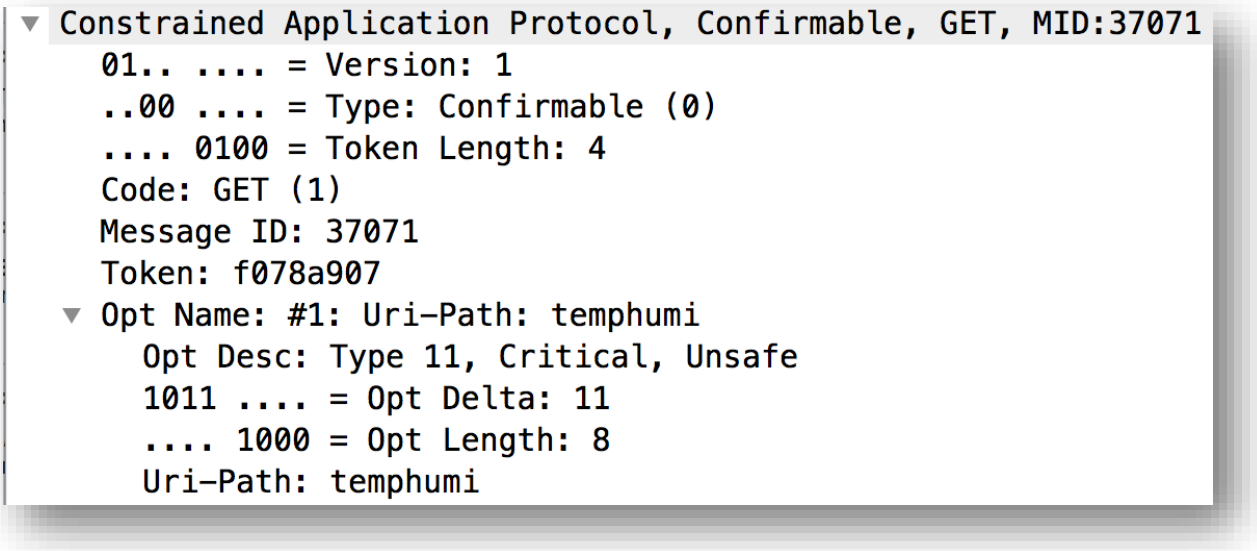

Figure 19 – The GET request dissected in Wireshark

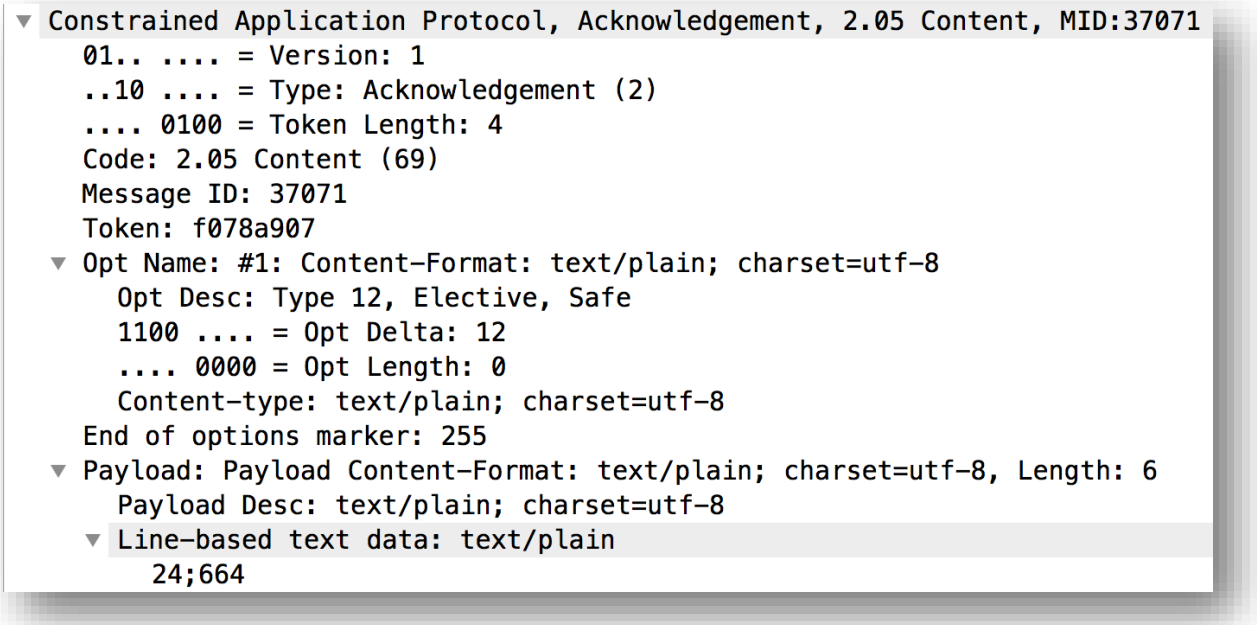

Figure 20 – A CoAP response ACK message dissected in Wireshark (values show temperature and humidity readings from the sensor)

3.4 Contiki OS

The easiest way to enable a wireless embedded device with 6LoWPAN is by integrating an existing protocol stack, either with a network processor, a stack included with an operating system or by integrating a stack into an embedded software project. A protocol stack for

6LoWPAN should ideally include, these basic components: radio drivers, medium access control, IPv6 with 6LoWPAN, UDP, ICMPv6, Neighbor Discovery and socket-like or other API to the stack. Two open source protocol stacks for embedded operating systems are popular today, uIPv6 for Contiki and BLIP for TinyOS, and three commercial protocol stacks: Sensinodes NanoStack, Jennics 6LoWPAN and the Nivis ISA100 stack [29]. Since, this work uses open source tools and recommendations, Contiki OS and Tiny OS were shortlisted as the operating systems for the research.

TinyOS is an open-source operating system developed for use in wireless embedded sensor network research [30]. TinyOS is designed for low-power embedded devices with limited amounts of flash and RAM, thus suitable for IoT applications. The OS uses a component-based structure and an event-driven execution model. Some object-oriented features are realized by utilizing a new language built upon C which is called NesC, which somewhat limits the portability of the operating system.

Contiki is called the open source operating system for the Internet of Things (IoT). It is a popular embedded open-source operating system for small microcontroller architectures such as AVR, 8051 and MSP430, led by the Swedish Institute for Computer Science (SICS) [31]. Contiki includes a very small implementation of IP called uIP [32], along with an implementation of IPv6 with 6LoWPAN support called uIPv6. The Contiki architecture is designed for supporting IP networking over low-power radios and other network interfaces. The operating system is implemented in C and uses a make build environment for cross-compilation on most platforms. Lots of reusable example implementations are available with good documentation to understand code structure.

The Contiki architecture is as shown in the Figure 24. The low-level hardware abstraction is split into platform and CPU for portability, which include hardware drivers. The Contiki OS provides basic thread and timer support. The Rime system is a flexible medium access control and network protocol library which includes many low-level communication paradigms. The uIPv6 stack makes use of Rime, and provides a socket-like API for use by applications called proto-sockets. Both built-in and user applications are run over Contiki using a lightweight thread model called proto-threads.

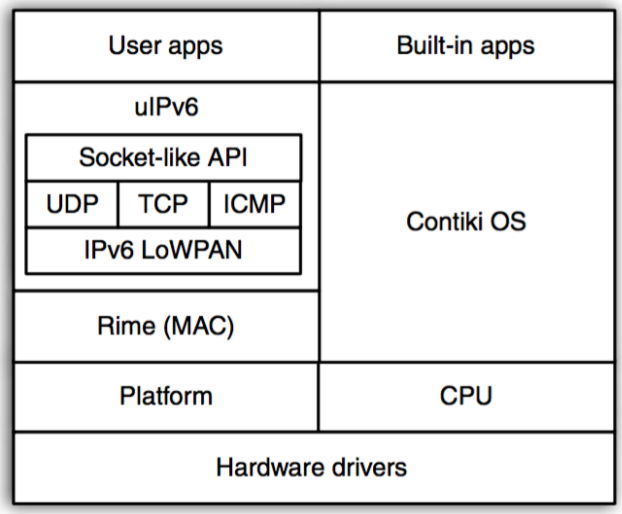

Figure 21 – The Contiki Architecture

For this work, the method of enabling 6LoWPAN was by using a stack with Contiki OS. Packets are inherently compressed using a 6LoWPAN adaptation layer by the Contiki OS kernel. Due to its simplicity, flexibility and greater availability, this research work employs Contiki OS as the default OS, for both simulations and physical deployment.

3.5 Californium (Cf) and the Copper client (Cu)

Californium is a powerful CoAP framework which implements CoAP in Java. It targets back end services and stronger IoT devices, providing a convenient API for RESTful Web services that support all of CoAP's features. The Californium (Cf) CoAP framework shows 33 to 64 times higher throughput than high-performance HTTP Web servers. The Californium reference implementation Cf also outperforms other CoAP systems with a three times higher throughput [33]. It is explicitly designed for scalable IoT cloud services and is publicly available at the Eclipse Foundation [34]. All these advantages motivated the use of this framework in the simulation scenarios and physical deployments for performance studies described in the next sections.

Attached below are some snapshots of the code where the Californium framework was employed in the application developed to send CoAP requests controlling different parameters such as inter frame time, number of requests etc. In the code below, the new URI is inserted using the prefix "coap://", this is followed by the sensor mote IP and the resources

exposed, in this case the temperature and humidity, shortened for ‘temphumi’. Then, a CoAP client gets instantiated with the URI, which contains the server’s IP, which in this case is the IP address of the sensor mote running the server. The CoAP request sender class then instantiates an object, with the server IP, delay desired between requests and the number of request messages desired to be sent. After sending the requests, the application waits for a response and when it receives one, further processing is done on it, else an unsuccessful reponse is acknowledged. Another factor, CoAP retransmit factor plays a part in this, by default the value is 4 for this factor, so the application waits for the response 4 times after sending requests, before deeming it as a failed response.

```
try{
  uri = new URI("coap://["+this.ipAddress+"]/temphumi");
 }
catch (URISyntaxException e) {
  System.err.println("Invalid URI: " + e.getMessage());
  System.exit(-1);
 }

  CoapClient client = new CoapClient(uri);

  CoapResponse response = client.get();
```

```
for(int i=0;i<noOfMote;i++){
    String IpAddress=this.prop.getProperty("mote"+(i+1));
    CoapRequestSender crs= new CoapRequestSender(IpAddress,delay
            ,noOfmsg,this);
    crs.start();
}
```

```
if (response!=null) {
    noOfSucSentMsg++;
    System.out.println(response.getCode()+" : "+response.getOptions()+" : "+response.getResponseText());
    System.out.println(response.advanced().getRTT());
    total_rtt += response.advanced().getRTT();
    avg_rtt = total_rtt / noOfSucSentMsg;

} else {
    noOfUnSucSentMsg++;
    System.out.println("No response received.");
}
```

Figure 22 – Snapshots of the Californium client application developed

The copper client Cu [35] is another tool developed by the eclipse foundation and it is available as a plugin for the Firefox browser. It allows browsing, bookmarking, and direct interaction with CoAP resources. This was a very useful tool while working with sensors for debugging as it would instantly tell whether the communication was working or not. Attached below is a screenshot with the temperature and humidity values from the sensor mote, directly accessed from the browser window using Copper in Firefox.

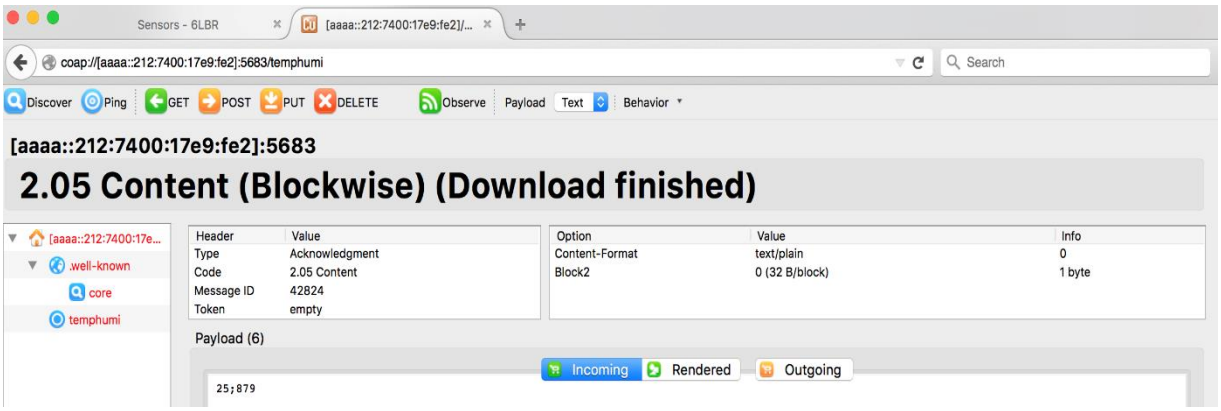

Figure 23 – Snapshot of the Copper plugin, reading sensor values

4. SIMULATIONS

Simulations do not suffer from NAT/ firewall issues, interconnectivity issues of IPv6/ IPv4, environmental interference and noise etc. They offer an ideal model of working which tries to simulate protocols as closely as possible to their recommendations and formats, to understand their behavior and structure. Different topologies and settings can be experimented with, to test the limits of networks.

4.1 Need for simulations

A simulator is a very useful tool for the application software development in wireless sensor networks. Before the simulator came up, the code development was very difficult and tedious for users due to the long compilation and debugs time and the transplant problem of the program. With the simulator tool, users can obtain much benefit during the software development phase and large scale test phase. But building simulations is not a trivial task, it involves building the right models/networks/topologies. Also, solid conclusions must be obtained from these simulations. Since real experiments prove to be too costly and time consuming, simulations pitch in as viable replacements. The testbeds built for smart city platforms such as Smart Santander, Smart Padova etc. are complex and expensive, thus making simulations a good alternative. Finally, simulations allow testing/validating new protocols and comprehensive benchmarking can be carried out.

4.2 Survey of different simulators

The various simulators researched for this work were OPNET, ns3, COOJA and OMNET++. OPNET although a powerful simulator, is mostly focused on traditional networks and did not have any toolboxes for IoT or WSNs. Ns3 is an open source network simulator which is a popular choice among a lot of open source projects and has support for 6lowpan networks with modules available for the same albeit it lacks a GUI and performance studies, obtaining results would have been more cumbersome by writing a lot of C++ code. OMNET++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators. It has an open source 6LoWPAN integration project associated with contiki-2.6, but it isn't straightforward integrating it as a wrapper over Contiki. Moreover, CoAP support still trails behind and with the introduction of Contiki 3.0, the code

structure was changed significantly enhancing reliability and optimizations in terms of memory. Cooja, being the default network simulator for Contiki came natively bundled along with Contiki 3.0. Cooja has a good GUI environment and allows for quick simulation setups and analysis. Cooja, therefore was found to be the best to simulate the network we have in mind, due to its flexibility, extensibility and quick prototyping.

4.3 Cooja

Cooja is a wireless sensor network simulator designed for the Contiki operating system. It is a flexible java based simulator which supports using C language to develop application software by Java Native Interface. One of the great advantages of this Cooja simulator is that it can simulate the application software simultaneously in high level algorithm development and low level hard driver development. The Cooja simulator has great extensibility. Application developer can alter parts of the simulation environment without changing any Cooja main code. It means that the system can be added new parts such as interfaces, plugins and radio mediums or reconfigured existing parts. With these advantages of Cooja, we can implement variant simulations with different conditions and system settings such as different packet generation rates, different MAC protocols and different network topology [36].

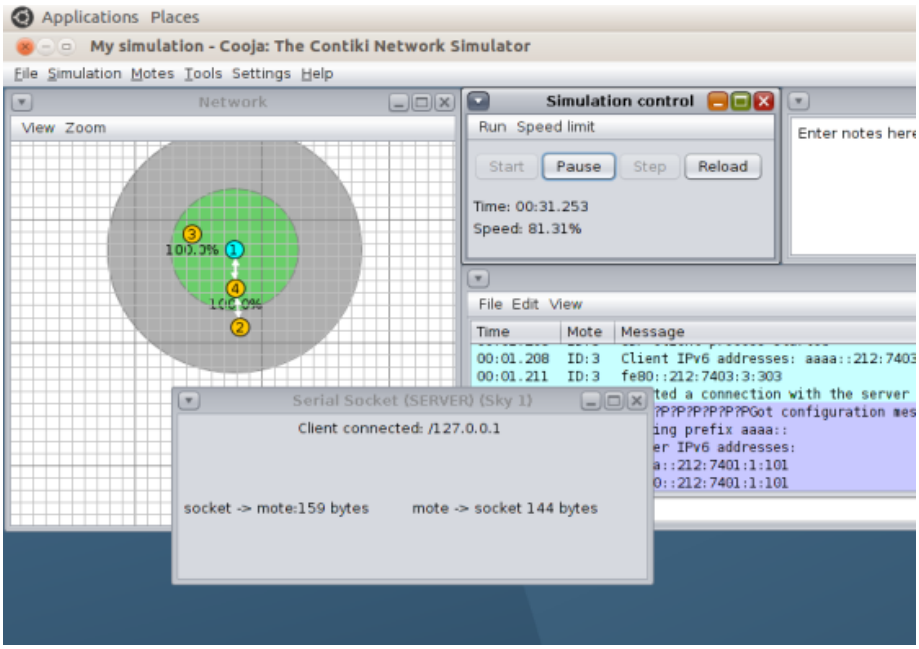

Figure 24 – The Cooja GUI environment

4.4 Simulation scenario

The simulation scenario is designed to test both the RPL routing as well as evaluate the performance of the stack. It consists of 5 motes running CoAP servers, providing temperature and humidity values as resources from the underlying SHT11 sensor. These motes are periodically queried by the CoAP client, Californium based Java program, which periodically sends GET requests to the servers in the simulation to obtain the current temperature and humidity values through the border router, which connects the network to the internet using the TUNSLIP utility, which helps to communicate between the border router and the external network. The network topology in Cooja is as shown in the Figure 25 below [37],

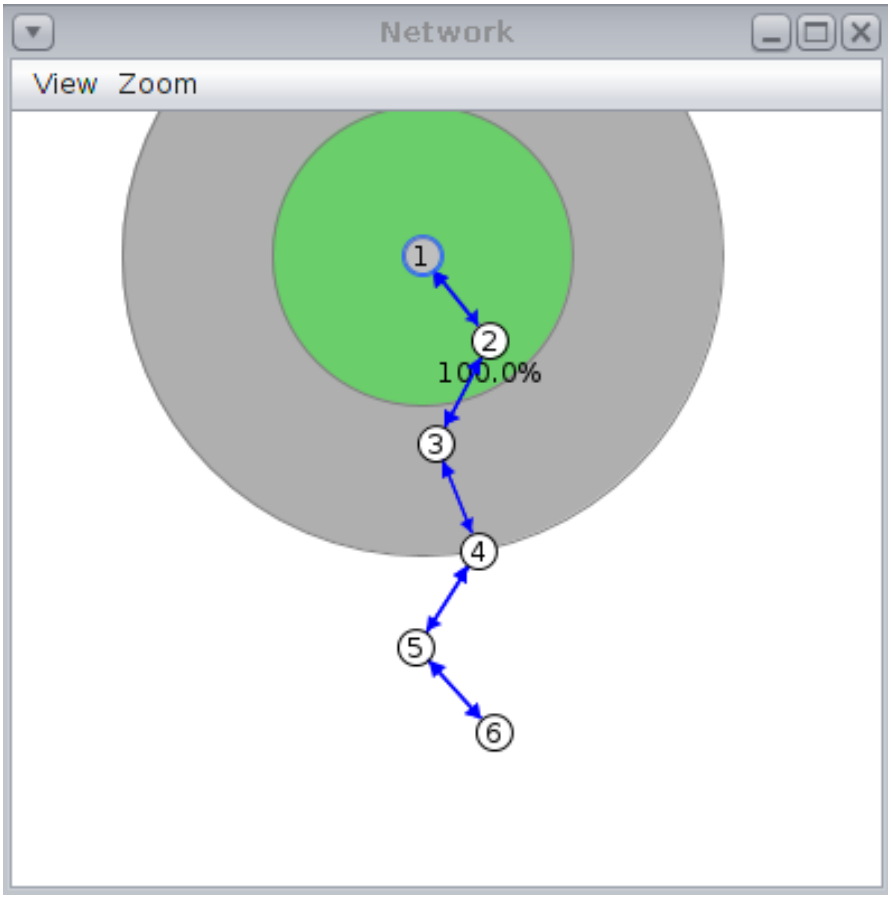

Figure 25 – Network topology in Cooja for the simulation scenario

The scenario is designed to have a multiple hop scenario, in which the motes running CoAP servers are positioned 1 to 5 hops away from the border router. These motes are then sent GET requests from the Californium application to obtain temperature and humidity values.

The general simulation parameters and their values are as shown below in the table,

Table 3 – General simulation parameters in Cooja

Parameter name	Value
Radio medium	Unit Disk Graph Medium
Mote Type/ startup delay	T-mote Sky/ 1000 ms
MAC layer	CSMA/CA
Bit rate	250 kbps
Radio duty cycling	NullRDC
Node transmission range	50 m
Node carrier sensing range	100 m
Tx / Rx ratio	100 %

The CoAP client used, Californium (Cf), can be customized and the tuned CoAP specific settings are as follows,

Table 4 – Californium properties

Property name	Value
Max retransmit factor	0
Max trasmit wait	93000
Ack timeout	2000
Ack random factor	1.5

4.5 Results

The scenario evaluates three important parameters namely the 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 [37].

4.5.1 Throughput

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 below,

Table 5 – Throughput variation (in Kbps) as a function of the packet generation rate, PGR (in packets/sec)

PGR	Throughput in Kbps				
	1-hop	2-hops	3-hops	4-hops	5-hops
1	0.8977	0.8635	0.7583	0.6904	0.6508
2	1.4425	1.292	1.02	0.8529	0.7129
5	2.1026	1.6152	1.203	0.8504	0.7424
10	2.45	1.6062	1.3027	0.8758	0.7832
20	2.098	1.3228	1.099	0.8065	0.6837
50	1.5733	0.9643	0.7021	0.5926	0.5464
100	1.2866	0.8082	0.5506	0.5126	0.4806

The graph below depicts the variation of the throughput for the various motes (in Kbps) at hops 1 to 5 against the packet generation rate in packets/sec,

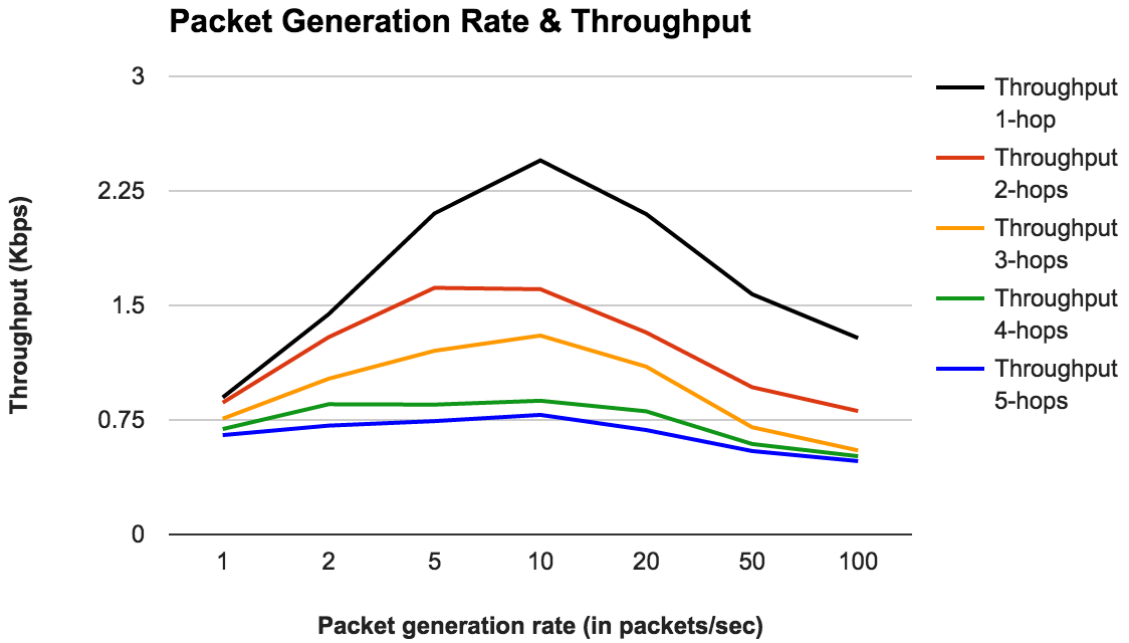

Figure 26 - Plot of average throughput for various motes at 1 to 5 hops against the rate of packet generation

The throughput is calculated as (in Kbps),
(No of successful CoAP request/response pairs * (length of request + length of response in bits)) / total time of simulation.
We can see from the graph that the throughput peaks at around 10 packets/sec and then declines. This can be attributed to the greater packet loss occurring due to increased incoming traffic.

4.5.2 End to end delay

The table below depicts the various values for the end to end delay in a similar simulation scenario varying the packet generation rate from 1 to 100 packets/sec,

Table 6 – End to end delay variation (in ms) as a function of packet generation rate (in packets/sec)

PGR	End to End delay in ms				
	1-hop	2-hops	3-hops	4-hops	5-hops
1	121.01	136.95	152.95	163.95	173.2
2	125.91	142.59	153.77	167.76	173.06
5	181.2	193.8	209.77	227.82	255.914
10	254.2	266.86	322.85	339.41	382.92
20	233.89	295.05	343.97	390.26	430.48
50	387.66	425.2	539.96	575.16	652.34
100	406.9	586.88	747.95	775.38	828.03

The end to end delay is an important parameter of evaluation as it is a direct indicator of the network congestion as well as the processing delays. It is computed as a function of the round trip time by the californium client Cf as the time difference between the time of the successful response reception for a particular request and the time instant at which the request is sent.

It is the sum of 2 * max latency and the processing delay.

The plot below illustrates a sort of almost linear relationship for the different hop scenarios,

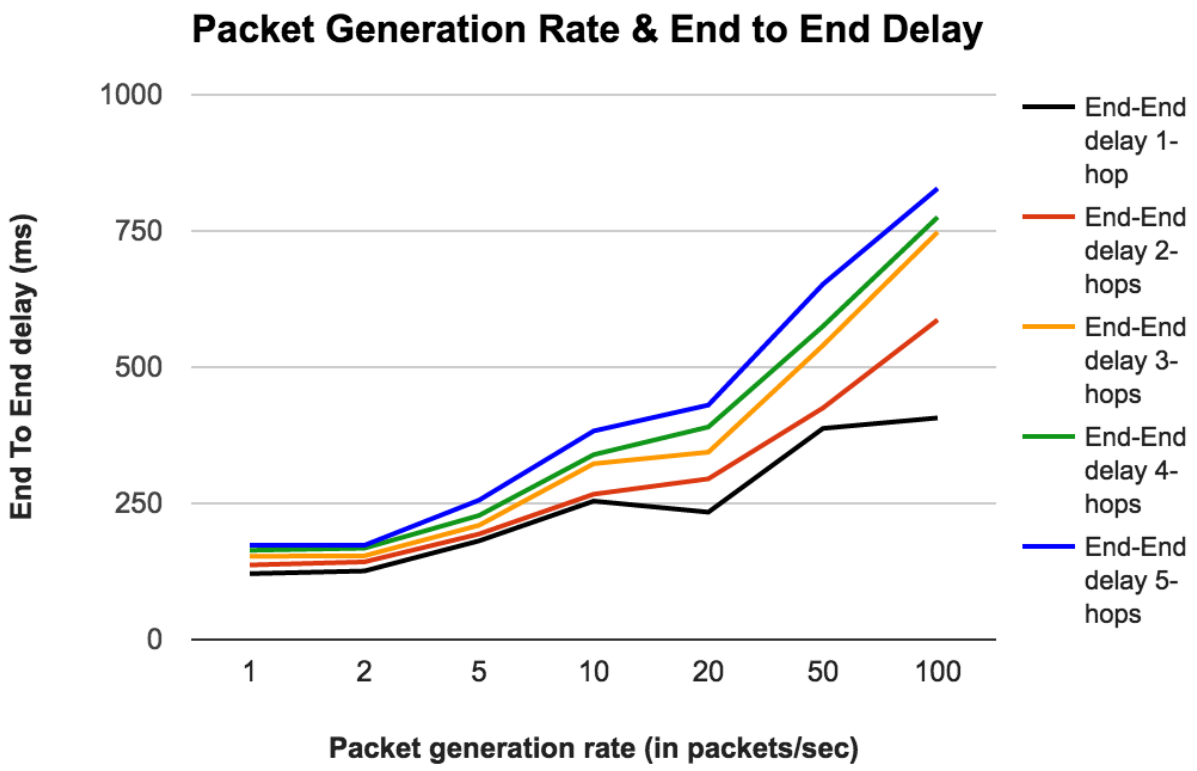

Figure 27 - Plot of end to end delay for various motes at 1 to 5 hops against the rate of packet generation

The plot shows a gradual increase in the end to end delay, almost linear beyond a certain point. Surprisingly there was not much difference between the PGR values of 10 and 20 packets/sec.

4.5.3 Packet loss

The GET requests sent by the CoAP client in the simulation scenario, which are not acknowledged determine the packet loss. Packet losses occur in environments where there is atmospheric noise present, 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,

Table 7 – Packet loss (in %) as a function of packet generation rate PGR (in packets/sec)

PGR	Packet loss in %				
	1-hop	2-hops	3-hops	4-hops	5-hops
1	1.8	2.7	6.7	9.8	11.7
2	3.7	5.8	11.6	16.4	21.4
5	4.4	9	14.9	23.3	26.7
10	3.1	7.2	18.7	22.9	25.3
20	8.1	15.4	20.5	25.6	29.7
50	9.6	20.7	27.5	32.7	33.9
100	14.2	22.3	31.8	33.9	35.3

This parameter is obtained directly from the CoAP client as the number of unsuccessful packet transmissions. The figure below illustrates this variation,

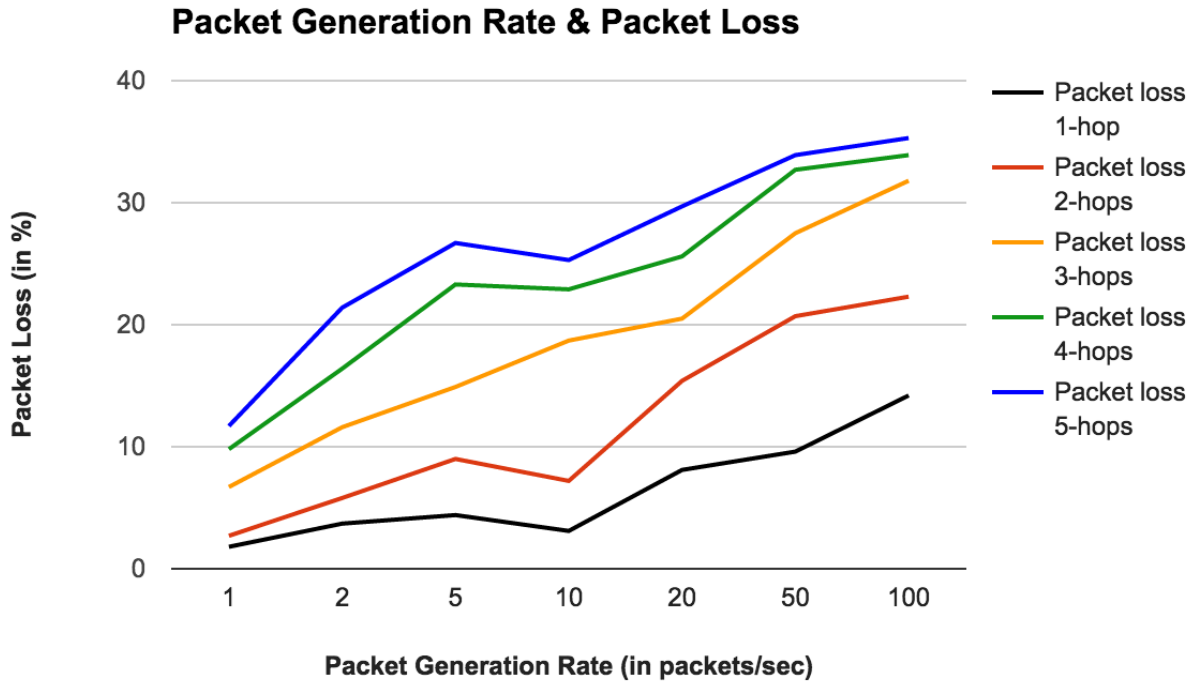

Figure 28 - Plot of packet loss for various motes at 1 to 5 hops against the rate of packet generation

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.

5. PHYSICAL DEPLOYMENT

CoAP exposes the sensor's resources as a server-controlled abstraction. Due to this, the gateway complexity reduces unlike when the application gateway exposes the sensor resources leading to higher complexity of implementation. When an application gateway requires to have knowledge of the functionalities of each connected device, flexibility is hampered in terms of design and scalability is limited. Non IP based standards like ZigBee suffer from this design flaw and the simple gateway design was enabled by the use of IP based stack in this work. Due to this, the gateways can be small devices as this design puts the emphasis on end nodes exposing their resources running CoAP servers.

5.1 Hardware platforms

Wireless sensors or motes as they are often referred to, are characterized by their small size, limited memory and processing power. The metrics while considering hardware for deployment include low cost, good range of operation, enough memory to operate applications, battery life etc. among others. The push for open standards in hardware where manufacturers do not restrict the motes' compatibility and functioning with other similar vendors is quintessential. IoT devices usually contain a microcontroller, a radio transceiver, flash memory and power source, typically battery operated. They also contain A/D converters as a peripheral. The more components that get integrated onto a device, the easier the deployment due to reduced interfacing to different components.

This work selected the hardware platforms, based on their compatibility with Contiki OS. The primary hardware platforms of Contiki, as on their website are Sensortag, CC 2538 and CC2650. The other platforms are as shown in the table below,

Table 8 – Hardware platforms supported by Contiki [31]

MCU/SoC	Radio	Platforms	Cooja support
TI CC2538	Integrated / CC1200	RE-Mote	-
RL78	ADF7023	EVAL-ADF7023DB	-
TI CC2538	Integrated	cc2538dk	-
TI MSP430x	TI CC2420	Exp5438, z1	Yes
TI MSP430x	TI CC2520	Wismote	Yes
Atmel AVR	Atmel RF230	Avr-raven, avr-rcb, avr-zigbit, iris	-
Atmel AVR	TI CC2420	Micaz	Yes
Freescale MC1322x	Integrated	Redbee-dev, redbee-econotag	-
TI MSP430	**TI CC2420**	**sky**	Yes
TI MSP430	TI CC1020	Msb430	-
TI MSP430	RFM TR1001	Esb	Yes
Atmel Atmega128 RFA1	Integrated	Avr-atmega128rfa	-
Microchip pic32mx795f512l	Microchip mrf24j40	Seed-eye	-
TI CC2530	Integrated	cc2530dk	-
6502	Integrated	cc2530dk	-
Native	-	Native, minimal-net, cooja	Yes

From the table, the different platforms were studied and the sky motes were found to be the most suitable for this work, due to their higher flash memory to run CoAP servers and relatively more documentation and availability of these modules. Within the sky platform there are further different motes present like XM1000, CM5000, CM5000SMA, CM3000, CM3300, CM4000 etc. The CM5000SMA was chosen for the presence of an external 5dBi antenna, which allowed for transmission range of 300 m outdoors. The XM1000 was also experimented with, as it possesses more mote memory and bigger applications specifically the csma mechanism at the MAC layer could be flashed onto it.

Next come the border routers, these are complex pieces of software integrating the WSN to the global internet and therefore require more memory and processing power. Border routers are vital in any deployment and often the most complicated to work with. They must also possess an Ethernet port and/or a Wi-Fi module to be connected to the global internet themselves. Sensor motes cannot be used for this purpose, due to memory and power constraints primarily, but also due to the fact that they lack an Ethernet/Wi-Fi module. For this purpose, the research work considered Raspberry Pi and Arduino Mega to function as border routers. The next section explains the specific scenarios that were thought of for deployment.

5.2 Deployment scenarios

Three physical deployment solutions are proposed below.

Scenario 1 -

Ease of deployment is an important metric when it comes to wireless sensor networks. Scenario 1 assumes lower complexity in terms of devices to be interfaced and the sensor motes are battery powered, both of which are important advantages. The sensor motes in this scenario are application specific, optimized in terms of battery efficiency and provide a range of 120 m outdoors. On the flip side though, is the price of these motes.

This scenario is a validation of our simulation scenario in the Cooja simulation environment using the Contiki OS. The hardware used is sky motes, running CoAP servers. Raspberry PI functions as an IoT gateway device connecting our sensor network to the Sense Smart City platform and eventually the web-based belief rule based system. This scenario is designed around the 6LBR project [38], 6LoWPAN border router solution.

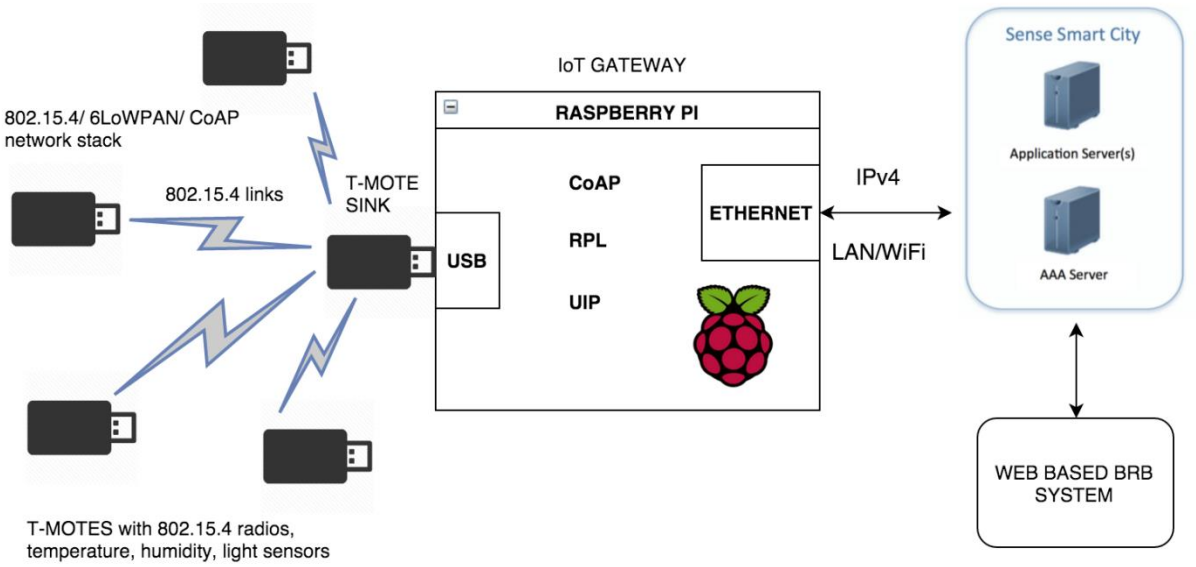

Figure 29 – Scenario using RPi as a border router, sky motes running CoAP servers and a sky mote as a slip radio connecting the IP and the wireless domains

Scenario 2 -

In this scenario, the popular open source hardware platform Arduino is employed. Low cost, low power Arduino Unos together with the temperature and humidity sensors and the Xbee series 1 radios function as sensor motes, relaying data using Arduino Mega as a border router. The ease of deployment is less in this case as different standalone components need to be interfaced, and made to work together. The hardware complexity is more and perhaps the biggest drawback is the power consumption as the Arduino is a multipurpose device, not designed only to operate in battery efficient conditions. Availability and price are good for this proposition. [39] serves as the inspiration for this scenario.

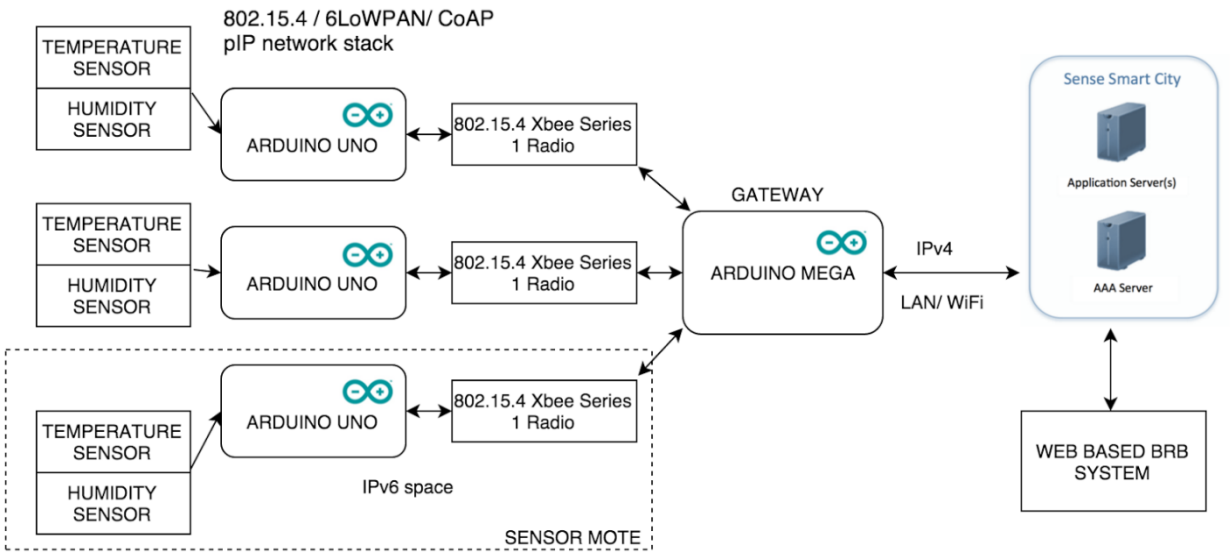

Figure 30 – Scenario with Arduino Unos interfaced with sensors running CoAP servers with xbee radio modules and Arduino Mega as a border router

Scenario 3 -

This scheme proposes Short messaging service (SMS) as a potential mean of transferring CoAP messages. This involves the usage of SMS - C (SMS service centers). Using CoAP over SMS as a transport level protocol is an interesting research premise and some basic performance analysis has been done for the same [40].

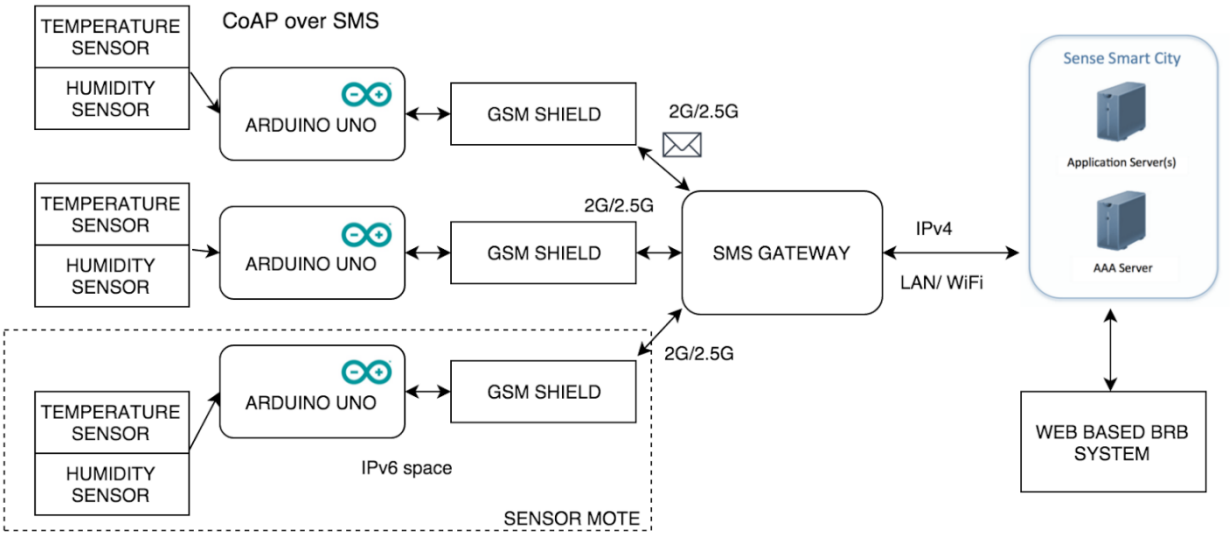

Figure 31 – Scenario using CoAP over SMS, Arduino based sensors, GSM shields and an SMS gateway over 2/2.5G technologies

The research work deployed scenario 1 as it served as a validation for the stack proposed as well as for the simulated stack in the earlier section. The ease of deployment is also in favor of this scenario, as the sky motes used have on chip 802.15.4 radios with antennas, SHT11 sensors with light, temperature and humidity data as well as a good amount of mote memory (48KB) and a TI MSP430F1611 Microcontroller. Among the various hardware motes, CM5000-SMA was chosen by comparison with other models on offer, also providing an external 5 dBi antenna, giving a range of ~300m(outdoor), 40~50m(indoor) according to specifications. Regarding the price, scenario 3 seems to be the cheapest, in terms of functionality and use of existing hardware, and makes a strong case for future use in different geographies. However, deployment presents problems, and availability of solutions seems low.

5.3 6LBR

CETIC 6LBR is a 6LoWPAN/RPL Border Router solution. 6LBR can work as stand-alone router on embedded hardware or on a Linux host. 6LBR is designed for flexibility, it can be configured to support various network topologies while smartly interconnecting the WSNs with the IP world. It runs out-of-the-box on low-cost and open hardware platforms and Linux hosts. The purpose of the 6LBR is to interconnect a WSN network, based on 802.15.4 and 6LoWPAN, with an existing IPv6 network and therefore was perfectly suited for this work.

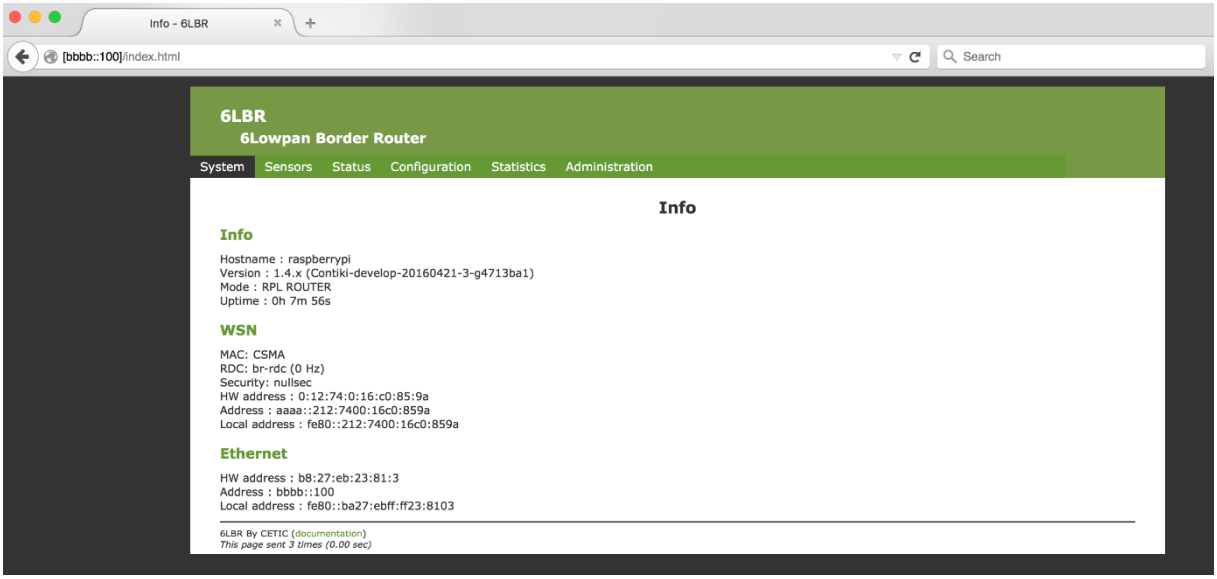

Figure 32 – The web browser of the 6LBR on Raspberry Pi

5.3.1 Modes of operation

6LBR runs in three categories of modes: bridge, router and transparent bridge. The router mode was used for this work. In this mode, the 6LBR acts as a full-fledged IPv6 Router, interconnecting two IPv6 subnets. The WSN subnet is managed by the RPL protocol and the Ethernet or Wi-Fi subnet is managed by IPv6 NDP. The 6LBR provides a virtual second interface to Contiki thanks to the packet filter module. This mode works more like a Gateway between Ethernet/Wi-Fi and the 6LoWPAN RPL. There are 4 different router modes in 6LBR – Router, NDP router, 6LR and RPL-Root. These can be changed using the 6lbr.conf file as shown in the appendix.

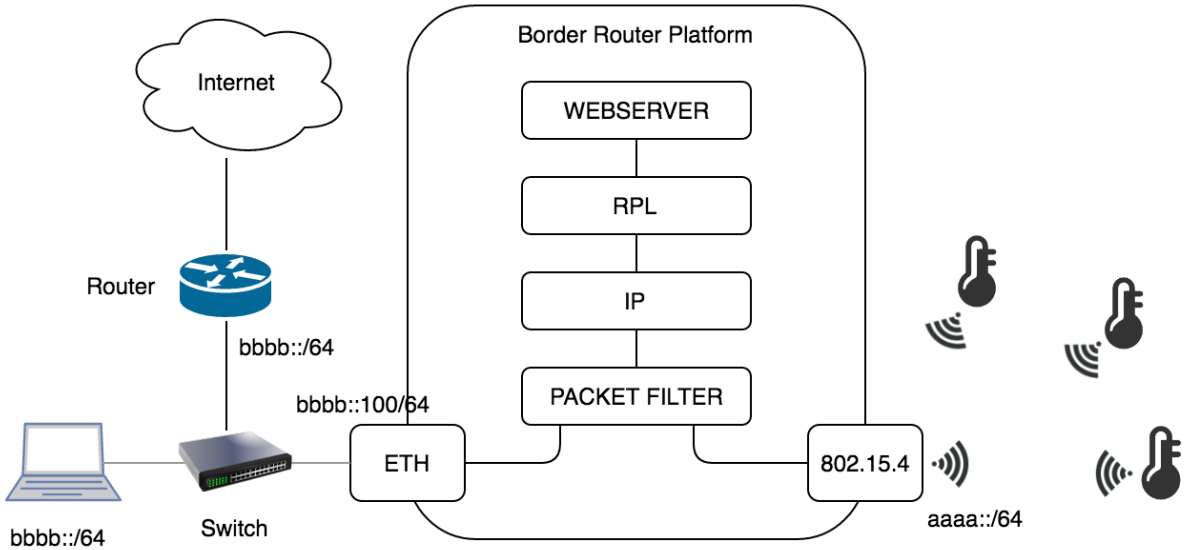

Figure 33 – The Router mode of operation of 6LBR

The 6LBR slip radio - When 6LBR is used on a Linux platform it requires an external piece of hardware running the 802.15.4 radio. On this hardware runs a dedicated firmware implementing the communication protocol between the 6LBR Linux process and the 802.15.4 radio. The data is transferred over UART or serial-usb. By default, the MAC layer is located inside 6LBR process and the RDC layer is located in the slip-radio. The default MAC is CSMA and the default RDC is nullrdc.

5.4 Experimental setup

The sensor motes are flashed with the CoAP server code in Contiki, the MAC is nullmac and the RDC is nullrdc. These sensor motes are then powered on by using AA batteries and places in the vicinity. One Sky mote is flashed with the slip-radio code, and this mote is directly connected to the Raspberry Pi via USB. The Raspberry Pi runs the 6LBR software on boot in the router mode as explained in the earlier section. Now the Pi is directly connected to the internet via Ethernet or Wi-Fi. The Californium application to send CoAP requests to these sensor motes is on a development PC. This development PC needs to be in the same subnet as the Pi or directly connected to the Pi via an Ethernet interface. The 6LBR software periodically sends a RA (router advertisement) to the subnet it is connected to via which the development knows the path to the sensors which are embedded in the advertisement. The default webserver is located at the address [bbbb::100] in the subnet which can now be accessed by the development PC, since it is in the same subnet. Now the sensors in the vicinity are discovered using the ICMPv6 and are shown in the web server (figure attached below for reference).

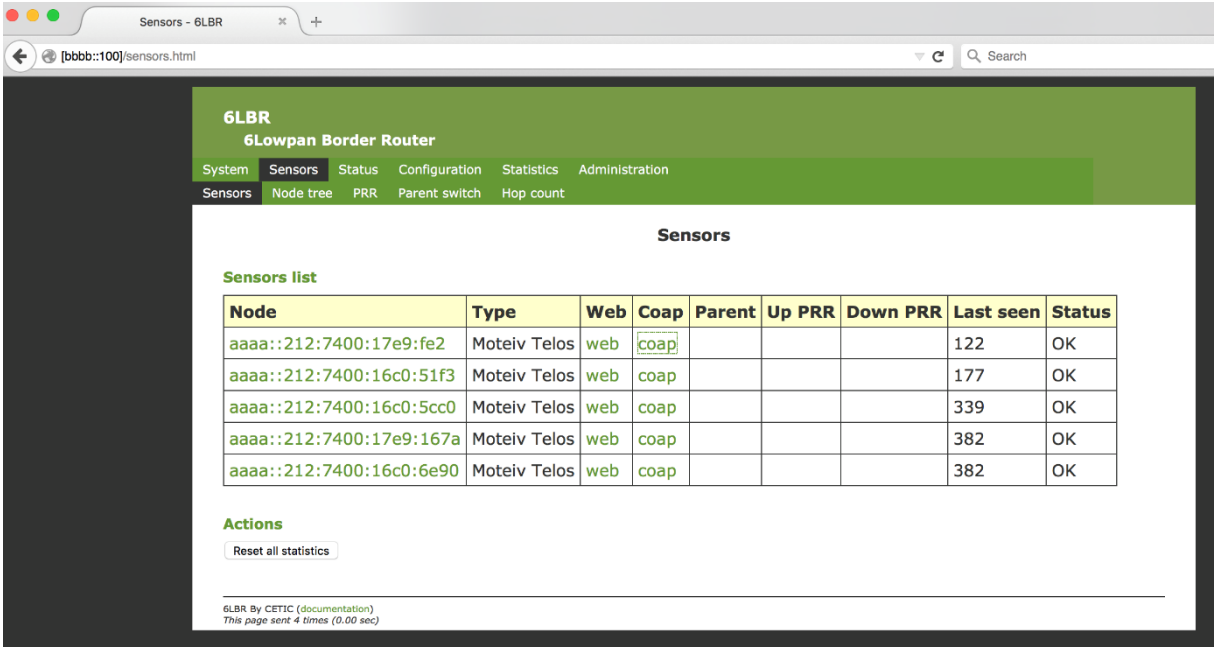

Figure 34 – The webserver showing the various sensors running CoAP servers

Now these sensors can be queried from the internet as they are globally connected via the border router 6LBR application running on the RPi. As a first step, they can be queried via the Copper plugin on firefox before the Californium application is launched.

Figure 35 – RPi running 6LBR, connected over USB to the slip radio and via Ethernet to a development PC and the various sensors running CoAP servers

5.5 Results

In the initial runs, it was found out that for an inter-request time of 10 ms, highest throughput was achieved. A scalability analysis was then carried out, to estimate the capacity of the 6LBR, slip radio and the sensor motes. With a setting of inter-request time of 10ms and the no of packets set to 1000, requests were launched and progressively motes running the CoAP servers were added to evaluate the parameters of throughput, delay and packet loss. The CoAP retransmit factor was set to 0 initially to intentionally get an accurate estimate of packet losses. These results are based on averaging over five runs to account for random behaviors.

Table 9 – Experimental setup parameters and values

Parameter	**Value**
Number of packets	1000
Inter-request time (delay)	10ms
No of motes	Varied from 1-5
CoAP retransmission factor	0
Slip-radio MAC	csma
CoAP server motes MAC/ RDC	nullmac, nullrdc
6LBR mode	Router

5.5.1 Throughput

The average throughput for all the sensors saw a sharp drop decline when even an additional mote was queried. The highest throughput achieved was 2.82 Kbps. The table below depicts the experimental throughput for each case as we add motes. The throughput is calculated as (in Kbps),

(No of successful CoAP request/response pairs * (length of request + length of response in bits)) / total time of simulation.

Table 10 – Average throughput (in Kbps) as a function of number of motes running CoAP servers

Number of motes	**Average Throughput (in Kbps)**
1	2.8114
2	1.7486
3	1.3614
4	1.2462
5	1.0426

Physical deployment throughput is not application layer throughput! The throughput is higher than simulations due to the additional addressing of the Pi, functioning as the border router, before getting to the slip radio and finally the WSN, giving more data to be transmitted.

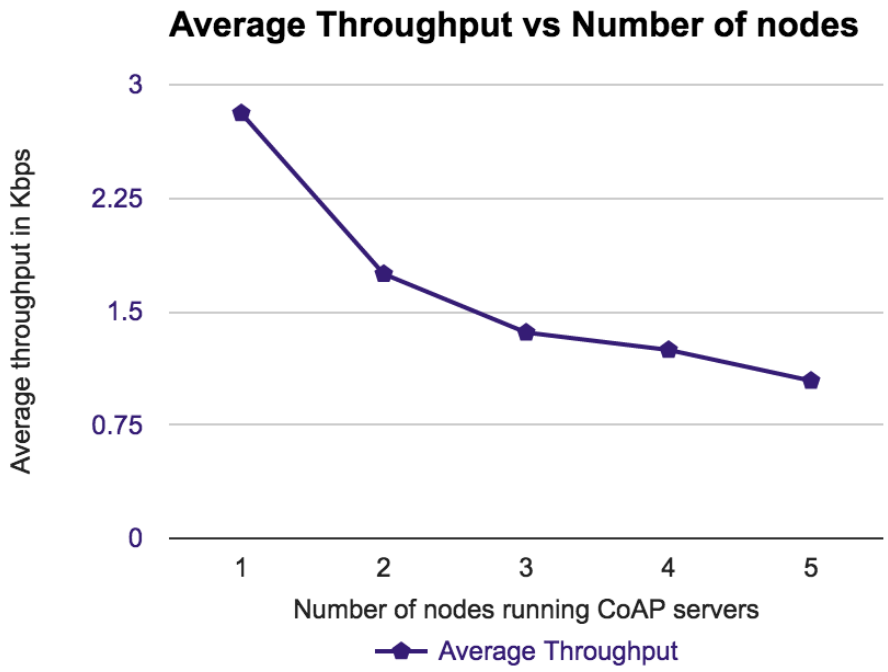

Figure 36 – Plot of average throughput (in Kbps) as a function of no of nodes/motes running CoAP servers

From the plot we can see a nonlinear drop when an additional mote is added, but thereafter it seems to have a constant linear decline. Still, with five motes we find the throughput is above 1 kbps and goes below the mark when a 6th node gets added. Therefore, for each border router and slip radio, if the application demands a throughput of > 1 Kbps, a maximum of 5 motes could be added in the network, beyond which it makes sense to have another border router with a slip radio.

5.5.2 End to end delay

In the experimental network, the end to end delay was an interesting parameter to study. This parameter is the sum of 2 * max latency and the processing delay and is obtained directly from the Californium application as round trip time.

The table below depicts the actual values that were recorded.

Table 11 – Average end to end delay (in ms) as a function of number of motes running CoAP servers

Number of motes	Average End to End delay (in ms)
1	416.628
2	426.76
3	437.09
4	433.48
5	530.772

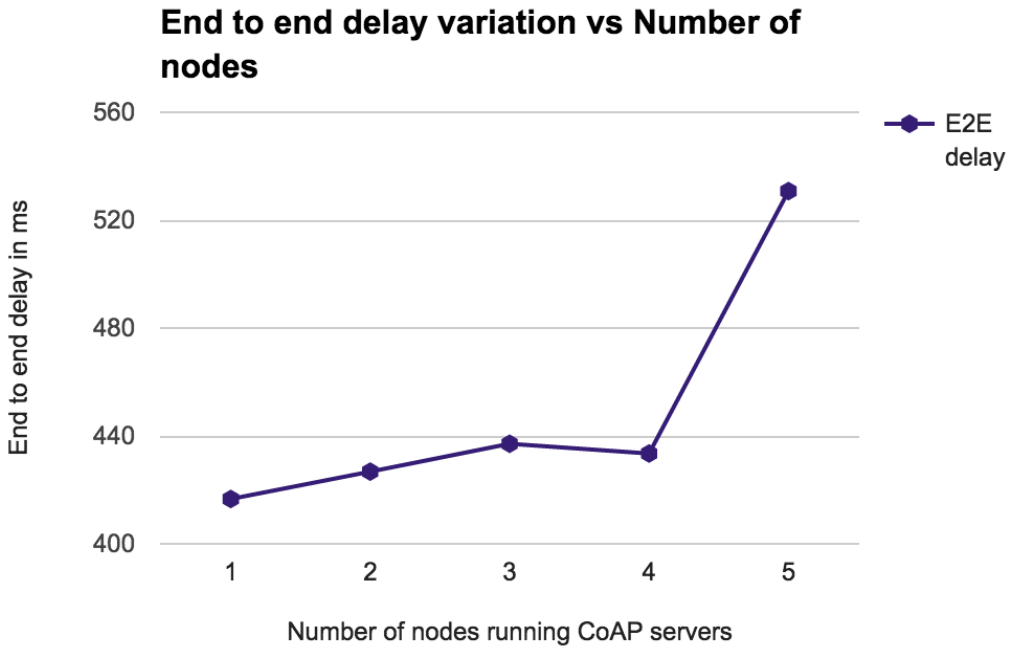

Figure 37 – Plot of average end to end delay (in ms) as a function of no of nodes/motes running CoAP servers

This parameter value remained around the 430 ms mark up to 4 motes and shot up to 530 ms when the 5th mote was added. This was a good observation in terms of latency requirements of the application although most real time applications would permit latencies up to a second.

5.5.3 Packet loss

CoAP requests from the Californium application, which are not acknowledged by the sensor motes determine the packet loss. The internal timer, RTO (retransmit timeout) times out and this factor gets incremented in the code. Since the retransmit is set to 0, it gives an accurate value of exactly how many packets were acknowledged. The table below shows the values that were observed,

Table 12 - Average packet loss (in %) as a function of number of motes running CoAP servers

Number of motes	Average Packet loss (in %)
1	0.7
2	9.65
3	16.5
4	18.725
5	21.34

Although the expectation was that this value would be very high, it was observed at worst to be 21.34 % for 5 motes, demonstrating the reliability of the stack over physical networks.

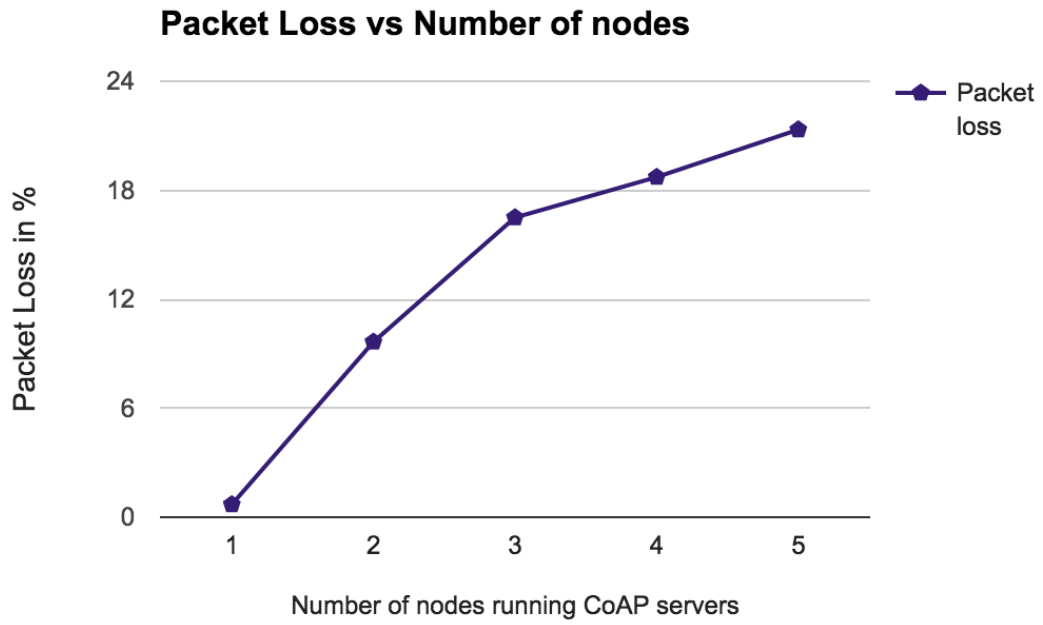

Figure 38 – Plot of packet loss (in %) as a function of no of nodes/motes running CoAP servers

The plot shows an almost linear increase in the packet loss as more motes get added to the network, although with one mote the packet loss is almost negligible.

However, the biggest testimonial to the reliability of the stack occurred when the CoAP retransmission factor was changed to 4, according to the RFC7252 [7] recommendation. There was zero packet loss even with 5 motes operating simultaneously to a border router and a slip radio. Furthermore, even when all the 1000 packets were sent at a time, the stack handled the onslaught of requests and the sensor motes replied to each one of the requests, thus making a strong case for reliability of the stack. The CoAP retransmission mechanism based on the RTO (retransmit timeout) thus, provides an efficient congestion control mechanism over UDP, which is unreliable. Also, when all 1000 packets were sent simultaneously, with the retransmit factor to 4, the end to end delay on average shot up to 1.2 seconds, which was understandable because of retransmissions with the same MID multiple times and the throughput stayed around 1 Kbps on average, with negligible packet loss.

6. DISCUSSION

The comparison of simulated and experimental stack indicates the following values in terms of 5 motes and a packet generation rate of 10ms,

Table 13 – Comparison of simulated and experimental values

Number of nodes 5, PGR 10ms	Average Throughput (in Kbps)	Average end to end delay (in ms)	Average packet loss (in %)
Simulated value	1.40358 Kbps	313.248 ms	15.44 %
Experimental value	1.0426 Kbps	530.772 ms	21.34 %

The table shows that the simulated values are better in terms of the performance metrics mentioned. This can be attributed to ideal conditions available in a simulator environment. The physical deployment values, although inferior, still are very promising and closely resemble the simulated values in the presence of noise and interference. The most important observation was that when the CoAP retransmission factor was tuned to a value of 4, negligible packet loss occurred for both simulations and physical deployments, demonstrating the reliability prowess of the stack.

The heterogeneous wireless sensor network stack, developed and demonstrated in this research is the key component of the flood detection application. Different open source hardware platforms, sensors, open network protocols and OS are employed to relay environmental data such as temperature, moisture, humidity, rainfall etc. from sensors and eventually feed it to a belief rule based decision support system. This research also, aside from developing and demonstrating the WSN stack, conducts performance analysis of the stack. This performance analysis helps in planning deployment and benchmarking and evaluating the viability of the stack for the flood detection application. The simulated and deployment values obtained for the flood detection application in terms of throughput, packet loss and latency allow the application to take into account the network characteristics and performance of the stack to come up with reliable evaluations and ensure an intuitive behavior. From the flood detection perspective, the deployment has to be carried out on large river banks with long perimeters hence performance & scalability analyses, like throughput,

latency, packet loss, no of motes, distance between motes etc. become crucial for proper deployment, which this research attempts to do. For the flood detection application, there needs to be acceptable levels of data loss in the presence of noise and reliable data transfers at modest data rates and delays. The experiments conducted in this research indicate very good values for these parameters as presented in the results section, ensuring the prowess of the chosen stack. The work addresses the research problems posed in terms of interoperability, simplicity, ease of deployment, reliability, low power consumption while keeping the deployment cost low for the flood detection application and allowing tiny resource constrained nodes to develop and demonstrate a flood detection application.

Contiki OS is an impressive operating system, offering many features and supporting the entire stack used for this work. Contiki OS supports multiple physical and link layer technologies including Wi-Fi, Ethernet, 802.15.4 etc., thus enabling the formation of a heterogeneous sensor network stack and is compatible with various hardware platforms as this work demonstrates. This stack is very simple to implement and easy to deploy as the Contiki kernel performs the 6LoWPAN adaptation natively and has support for CoAP.

6LoWPAN potentially overcomes the interoperability problem. In the case with ZigBee, bridging between ZigBee and non-ZigBee needs more complex gateways unlike 6LoWPAN. The interoperability prowess of 6LoWPAN emerges as the single biggest advantage over ZigBee as the simulation and experiments demonstrate the communication between the motes in the network and applications outside the network using CoAP, employing nothing else than a normal router. The network operation is also demonstrated for heterogeneous networks in different address spaces, like in this case the Californium application is in IPv4 whereas the CoAP servers reside in an IPv6 address space.

Unlike HTTP, CoAP is built on top of UDP and has a compact packet overhead. This has a considerable impact on the energy consumption and response time of the motes. This is quantified by the simulation scenario and the physical deployment where a CoAP server run on a temperature and humidity sensor is sent GET requests which are 13 bytes whereas a CoAP response is just 17 bytes. In addition, we demonstrated that CoAP is not particularly sensitive to the increase of client request generation time. This is an important advantage because the application of flood detection requires frequent access to WSN data.

CoAP also allows the tuning of parameters like the CoAP re-transmission factor, which can be customized to the use case and application scenario. CoAP with its congestion control mechanism based on the request time out (RTO) and re-transmission factor also ensures reliability, we observed that for a default CoAP re-transmission factor of 4, even when all 1000 packets are sent all at once and the scenario of 5 hops, negligible packet losses occurred in the simulation. In the physical deployment, the performance was even better where there was not a single packet lost even when 5 sensor motes were added in the network. This is an important advantage of CoAP, which implements congestion control mechanisms over the unreliable transport layer protocol UDP and a testimonial to the reliability of the stack.

The stack simulated in this work is interoperable as it can be connected to different mote platforms and doesn't require proprietary gateways to connect to the internet. Cooja enables emulation with real hardware motes which makes the network simple and re-usable, 6LBR provides an easy to deploy border router solution for physical networks. The physical deployment will cost around 650 Euros for a system up to 5 sensor motes and the raspberry pi as a border router, but since it is battery powered, the lifetime of the application is expected to be some years. Therefore, the simulations and physical deployments demonstrate and validate the suggested stack as an optimal fit for real time applications like flood detection.

7. CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

The stack deployed is interoperable, extremely reliable as demonstrated by deployments and scalable while providing good throughput, acceptable latency and packet loss as seen from the results. It uses open standards and technologies, open hardware platforms and is prototyped at a low cost. The system of a border router and 6 sensor motes could theoretically cover a distance of about 2 kms and would cost around 650 Euros. The motes are battery powered and the system would last a few years. They are also accessible throughout the internet, enabling open access to sensor values in real time.

The research work will be useful in future research contexts related to the IoT and Wireless Sensor Networks, allowing developers to concentrate more on their application use cases with comprehensive benchmarks and performance analysis of stacks available, both in terms of simulations and real deployments. The work will,

- help understand the infrastructure requirements of IoT applications simplifying the development process,
- help promote open standards, IP based stacks, synergizing efforts and reducing proprietary technologies fostering interoperability,
- help design the physical deployment effectively in terms of how many hops are supported and positioning of base station/router depending on the application use case,
- analyze throughput, latency and packet loss requirements for the real-time applications.

This research work both directly and as a consequence promotes sustainability,

- by reducing vendor lock-in, producing compatible hardware and obsoleting application gateways.
- As a consequence, by building the flood detection application invaluable to the three pillars of economy, ecology and equity.

7.2 Future work

6LoWPAN adopts a mesh topology and uses a routing algorithm which does not take care of the sleeping mode thus requiring approaches such as low-power listening for energy saving purpose. Such energy saving modes will be an engaging area of research.

The Contiki power profiler and experimentation with different RDC layers and MAC layers such as csma, nullmac, contikimac etc. also would be a useful area of study. The memory requirement of the CoAP server programs at the moment is too big for sky motes but with the introduction of XM1000 modules with a 116 KB flash memory, the network behavior and performance analysis would be interesting to note for these different RDC/MAC layers.

In addition to this, computing the re-transmission time out (RTO) in CoAP is an interesting research question, which helps determine better congestion control mechanisms for such stacks. This factor is left open in the RFC 7252, and various CoAP implementations differ in the usage of this timeout factor computation.

In the physical deployment, the 3rd scenario of using CoAP over SMS seems a promising area to be explored, as it can be applied to different geographies and even better, at lower prices of deployment. Finally, the physical deployment scenarios could be compared with cost, reliability, energy efficiency, simplicity etc. metrics to determine the best hardware platforms and designs for various application use cases.

REFERENCES

1) IEEE, The Institute, Special Report: The Internet of Things, 2014, Available - http://theinstitute.ieee.org/static/special-report-the-internet-of-things
[Retrieved on May 12th, 2016].
2) IEEE P2413 is available at,
http://standards.ieee.org/develop/project/2413.html
[Retrieved May 12th, 2016].
3) Lee, G.M., Park, J., Kong, N., and Crespi, N. 2011. The internet of things: concept and problem statement: 01.
Available - https://tools.ietf.org/id/draft-lee-iot-problem-statement-00.txt
[Retrieved on May 12th, 2016].
4) Akyildiz, I.F. and Vuran, M.C. 2010. *Wireless sensor networks* (Vol. 4). John Wiley & Sons.
5) The ZigBee alliance, Available - http://www.zigbee.org/,
[Retrieved on May 16th, 2016].
6) Montenegro, G., Kushalnagar, N., Hui, J., and Culler, D. 2007. Transmission of IPv6 packets over IEEE 802.15. 4 networks. *Internet proposed standard RFC*, *4944*.
7) Shelby, Z., Hartke, K., and Bormann, C. 2014. The Constrained Application Protocol (CoAP). *Internet proposed standard RFC*, *7252*.
8) Kitchenham, B. 2004. Procedures for performing systematic reviews. *Keele, UK, Keele University*, *33*(2004), pp.1-26.
9) White paper, Zigbee Vs 6LoWPAN for sensor networks, LSR. Available at, https://www.lsr.com/white-papers/zigbee-vs-6lowpan-for-sensor-networks,
[Retrieved on May 17th, 2016].
10) The Eclipse Newsletter, CoAP and MQTT, IoT protocols, Available at, https://eclipse.org/community/eclipse_newsletter/2014/february/article2.php,
[Retrieved on May 17th, 2016].
11) Colitti, W., Why ZigBee has failed in IoT interoperability, http://www.waltercolitti.consulting/analysis/why-zigbee-has-failed-in-iot-interoperability/,
[Retrieved on April 27th, 2016].

12) Mulligan, G. 2007. The 6LoWPAN architecture. In *Proceedings of the 4th workshop on Embedded networked sensors* (pp. 78-82). ACM.

13) Astorga, J., Aguado, M., Jacob, E., and Matias, J. 2014. Evaluating the Viability of IPv6-Based Communications Over IEEE 802.15.4 Networks Using 6LoWPAN, 2014.

14) Colitti, W., Steenhaut, K., and De Caro, N. 2011. Integrating wireless sensor networks with the web. *Extending the Internet to Low power and Lossy Networks (IP+ SN 2011)*.

15) Kushalnagar, N., Montenegro, G., and Schumacher, C. 2007. *IPv6 over low-power wireless personal area networks (6LoWPANs): overview, assumptions, problem statement, and goals* (No. RFC 4919). RFC 4919 (Informational), Internet Engineering Task Force.

16) Thombre, S. 2015. Belief rule based DSS - flood assessment using WSNs, Geomundus, Lisbon, 2015.

17) Willows, R., Reynard, N., Meadowcroft, I., and Connell, R. 2003. *Climate adaptation: Risk, uncertainty and decision-making. UKCIP Technical Report*. UK Climate Impacts Programme.

18) Andersson, K. and Hossain, M.S. 2014. Smart risk assessment systems using belief-rule-based DSS and WSN technologies. In *Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE), 2014 4th International Conference on* (pp. 1-5). IEEE.

19) Yang, J.B., Liu, J., Wang, J., Sii, H.S., and Wang, H.W. 2006. Belief rule-base inference methodology using the evidential reasoning approach-RIMER. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *36*(2), pp.266-285.

20) Andersson, K. and Hossain, M.S. 2015. Heterogeneous wireless sensor networks for flood prediction decision support systems. In *Proceedings of the 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (pp. 133-137).

21) The IPSO alliance, Available - http://www.ipso-alliance.org/, [Retrieved on May 16th, 2016].

22) Klas, G., Rodermund, F., Shelby, Z., Akhouri, S., and Hller, J. 2014. Lightweight M2M": Enabling Device Management and Applications for the Internet of Things. *White Paper from Vodafone, Ericsson and ARM*, *26*.

23) IEEE Std 802.15.4 .2011. IEEE Standard for Local and metropolitan area networks - Low-Rate Wireless Personal Area Networks (LR-WPANs), (Revision of IEEE Std 802.15.4-2006), Sept, 2011.

24) Shelby, Z., Chakrabarti, S., and Nordmark, E. 2011. Neighbor discovery optimization for low power and lossy networks (6LoWPAN), *IETF draft-ietf-6lowpan-nd-21*.

25) Conta, A. and Gupta, M. 2006. Internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification.

26) Winter, T. 2012. RPL: IPv6 routing protocol for low-power and lossy networks.

27) Hartke, K. 2015. Observing Resources in the Constrained Application Protocol (CoAP). *Internet proposed standard RFC*, *7641*.

28) Dunkels, A., Österlind, F., Tsiftes, N., and He, Z. 2007. Software-based sensor node energy estimation. In *Proceedings of the 5th international conference on Embedded networked sensor systems* (pp. 409-410). ACM.

29) Shelby, Z. and Bormann, C. 2011. *6LoWPAN: The wireless embedded Internet* (Vol. 43). John Wiley & Sons.

30) The TinyOS Community, Available Wiki page - http://tinyos.stanford.edu/tinyos-wiki/index.php/TinyOS_Overview,
[Retrieved on 21st May, 2016].

31) Contiki: The open source operating system for the internet of things. Available - http://www.contiki-os.org/index.html,
[Retrieved on December 9th, 2015].

32) Dunkels, A. 2003. Full TCP/IP for 8-bit architectures. In *Proceedings of the 1st international conference on Mobile systems, applications and services* (pp. 85-98). ACM.

33) Kovatsch, M., Lanter, M., and Shelby, Z. 2014, Californium: Scalable Cloud Services for the Internet of Things with CoAP, International Conference on the Internet of Things, IEEE, 2014. Available - http://www.eclipse.org/californium/,
[Retrieved on January 4th, 2016].

34) Eclipse Foundation, Californium, Available at http://www.eclipse.org/californium/,
[Retrieved on December 20th, 2015].

35) Kovatsch, M., Copper (Cu) CoAP user-agent for Firefox, Plugin for Firefox available - https://addons.mozilla.org/en-US/firefox/addon/copper-270430/,
[Retrieved on 19th December, 2015].

36) Saad, L.B., Chauvenet, C., and Tourancheau, B. 2011. Simulation of the RPL Routing Protocol for IPv6 Sensor Networks: two cases studies. In*International Conference on Sensor Technologies and Applications SENSORCOMM 2011*. IARIA.

37) 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.

38) The 6LoWPAN border router, 6LBR, Available at https://github.com/cetic/6lbr/wiki [Retrieved 20th March, 2016].

39) The Arduino micro-IPv6 Stack for Arduino and Xbee based on Contiki OS, Telecom Bretagne, https://github.com/telecombretagne/Arduino-IPv6Stack/, [Retrieved on 20th March, 2016].

40) Gligoric, N., Dimcic, T., Drajic, D., Krco, S., Dejanovic, I., Chu, N., and Obradovic, A. 2012. CoAP over SMS: Performance evaluation for machine to machine communication. In *Telecommunications Forum (TELFOR), 2012 20th* (pp. 1-4). IEEE.

Appendix A - Software repositories and files

1. Project files - https://github.com/sumeet9958/CoojaCfSimulations

2. Contiki OS - https://github.com/contiki-os/contiki
   Wiki link for Installation, build, simulations, community, platforms etc.,
   https://github.com/contiki-os/contiki/wiki

3. 6LBR - https://github.com/cetic/6lbr
   Wiki link for installation on raspberry pi/beagle bone –
   https://github.com/cetic/6lbr/wiki/Other-Linux-Software-Configuration

4. Californium (Cf) CoAP Framework - https://github.com/eclipse/californium

5. Copper (Cu) CoAP agent - https://github.com/mkovatsc/Copper
   Plugin for Firefox available - https://addons.mozilla.org/en-US/firefox/addon/copper-270430/

Appendix B - Californium application properties and settings

Californium.properties – Californium settings and configuration file

#Californium CoAP Properties file
#Thu May 12 08:10:56 CET 2016

DEFAULT_ENDPOINT_THREAD_COUNT=1
MAX_TRANSMIT_WAIT=93000
MAX_RETRANSMIT= 0 # or 4 for reliability tests
UDP_CONNECTOR_SENDER_THREAD_COUNT=1
DEFAULT_LEISURE=5000
NOTIFICATION_MAX_AGE=128000
CROP_ROTATION_PERIOD=2000
USE_RANDOM_MID_START=true
ACK_TIMEOUT_SCALE=2
UDP_CONNECTOR_DATAGRAM_SIZE=2000
DEDUPLICATOR=DEDUPLICATOR_MARK_AND_SWEEP
HTTP_PORT=8080
NSTART=1
NOTIFICATION_CHECK_INTERVAL=86400000
MAX_MESSAGE_SIZE=1024
DEFAULT_COAP_PORT=5683
DEFAULT_BLOCK_SIZE=512
UDP_CONNECTOR_RECEIVE_BUFFER=0
ACK_TIMEOUT=2000
HTTP_CACHE_SIZE=32
UDP_CONNECTOR_OUT_CAPACITY=2147483647
HTTP_SERVER_SOCKET_BUFFER_SIZE=8192
ACK_RANDOM_FACTOR=1.5
MARK_AND_SWEEP_INTERVAL=10000
NOTIFICATION_CHECK_INTERVAL_COUNT=100
HTTP_CACHE_RESPONSE_MAX_AGE=86400
UDP_CONNECTOR_LOG_PACKETS=false
PROBING_RATE=1.0
NOTIFICATION_REREGISTRATION_BACKOFF=2000
EXCHANGE_LIFECYCLE=247000
SERVER_THRESD_NUMER=1
UDP_CONNECTOR_RECEIVER_THREAD_COUNT=1
UDP_CONNECTOR_SEND_BUFFER=0
HTTP_SERVER_SOCKET_TIMEOUT=100000
USE_RANDOM_TOKEN_START=true

Appendix C – 6LBR configuration and settings

Path = /etc/6lbr/6lbr.conf – configuration file for the 6LBR

```
# Ethernet / Wi-Fi Configuration
MODE= ROUTER            # Runtime mode of the 6LBR

RAW_ETH= 1              # Raw mode, existing Ethernet interface
DEV_ETH= eth0           # wlan0 if Wi-Fi is used, both work
RAW_ETH_FCS= 0
DEV_TAP= tap0           # unused in router mode
BRIDGE= 0               # no interface bridging
DEV_BRIDGE= br0         # unused in router mode

# Slip radio configuration
DEV_RADIO= /dev/ttyUSB0         # where the slip radio is connected
BAUDRATE= 115200

# Logging
LOG_6LBR= /var/log/
LOG_6LBR_OUT= /var/log/6lbr.log
LOG_6LBR_ERR= /var/log/6lbr.err
LOG_LEVEL= 3

# 6LBR global Ethernet interface address
IP_CONFIG_FILE= /var/log/6lbr.ip                # Webserver access IP
```