IP based Wireless Sensor Networks: Performance Analysis using Simulations and Experiments

Sumeet Thombre[1], Raihan Ul Islam[1], Karl Andersson[1*], and Mohammad Shahadat Hossain[2]
[1]*Pervasive and Mobile Computing Laboratory*
Luleå University of Technology
SE-931 87 Skellefteå, Sweden
{sumeet.thombre, raihan.ul.islam, karl.andersson}@ltu.se
[2]*Department of Computer Science and Engineering*
University of Chittagong, Chittagong-4331, Bangladesh
hossain_ms@cu.ac.bd

Abstract

Wireless sensor networks are at the crux of the Internet of Things applications. At the current state, there exist several technologies competing against each other in the IoT space. These proprietary technologies and hardware pose a serious problem of interoperability, which is vital to unleash the vision of the Internet of Things. Moreover, the traditional approach towards wireless sensor networks was to be unlike the internet, primarily because of the power and memory constraints posed by the tiny sensor nodes. The IETF 6LoWPAN technology facilitates the usage of IPv6 communications in sensor networks, which helps solve the problem of interoperability, enabling low power, low cost micro-controllers to be globally connected to the internet. Another IETF technology, CoAP allows interactive communication over the internet for these resource constrained devices. Along with 802.15.4, 6LoWPAN and CoAP, an open, standardized WSN stack for resource constrained devices and environments becomes available. The Contiki OS, touted as the open source OS for IoT, provides low power IPv6 communications and supports the 6LoWPAN and CoAP protocols, along with mesh routing using RPL. Along with these, a CoAP framework, Californium (Cf) provides a scalable and RESTful API to handle IoT devices. These open tools and technologies are employed in this work to form an open, inter-operable, scalable, reliable and low power WSN stack. This stack is then simulated using Contiki's default network simulator Cooja, to conduct performance analysis in varying conditions such as noise, topology, traffic etc. Finally, as a proof of concept and a validation of the simulated stack, physical deployment is carried out, using a Raspberry Pi as a border router, which connects the wireless sensor network to the global internet along with the T-mote sky sensor motes. Therefore, this work develops and demonstrates an open, interoperable, reliable, scalable, low power, low cost WSN stack, both in terms of simulations and physical deployments, and carries out performance evaluation of the stack in terms of throughput, latency and packet loss.

Keywords: The Internet of Things, Wireless Sensor Networks, 6LoWPAN, CoAP, Contiki OS.

1 Introduction

The term the Internet of Things has no specific definition. It is promulgated as a buzz word in research and industry today, and has generated a lot of attention in the sectors of city infrastructures, health care, transport, logistics etc. 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].

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 7:3 (September 2016), pp. 53-76
*Corresponding author: Tel: +46-(0)910-585364

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.

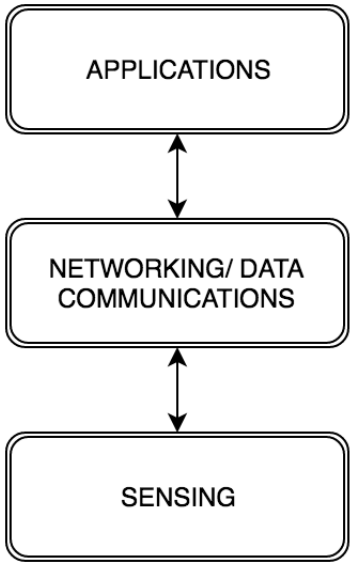

Figure 1: Three tiered IoT architecture

The Internet Engineering Task Force, IETF, provides its own description of the Internet of Things,

"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 must be designed with sophisticated and extremely efficient communication protocols along with sensors with advanced sensing capabilities. This paper 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. This work also 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 proliferation of 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 work. With so many tools and technologies available, the first step of this work was to examine the smorgasbord of possibilities. 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 becomes quintessential for the development of IoT applications.

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 applications' deployment accordingly. The work uses open source tools, operating systems, hardware platforms and technologies, and the code of this work is made publicly available and accessible.

2 Related work

2.1 IP in WSNs

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 [4] 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 WSNs and pushing it to the very edge of the network devices flattens the naming and addressing hierarchy and thereby simplifies the connectivity model [5]. 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.

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.

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.

2.2 IoT protocols

The current state of IoT presents a smorgasbord of protocols and technologies. 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 [6] 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.

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 [7] 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.

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 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 [8].

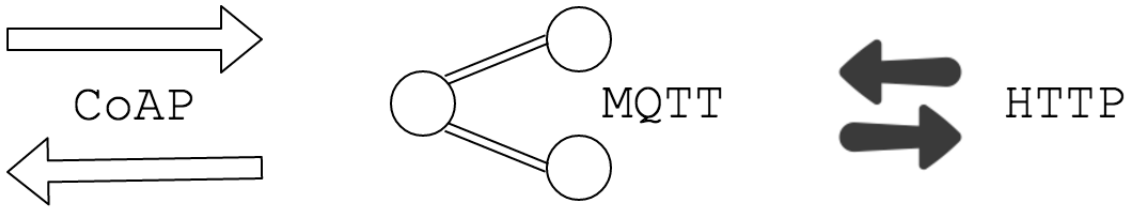

Figure 2: IoT application layer protocols

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. 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. Congestion control mechanisms are important for event based WSNs and [9] talks about an approach for the same using TDMA. [10] talks about PEER, an energy efficient routing algorithm for WSNs, which increases node lifetime with reduced energy consumption.

2.3 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 [11]. 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 percent 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.

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 [12].

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.

2.4 Belief rule based systems

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 [13].

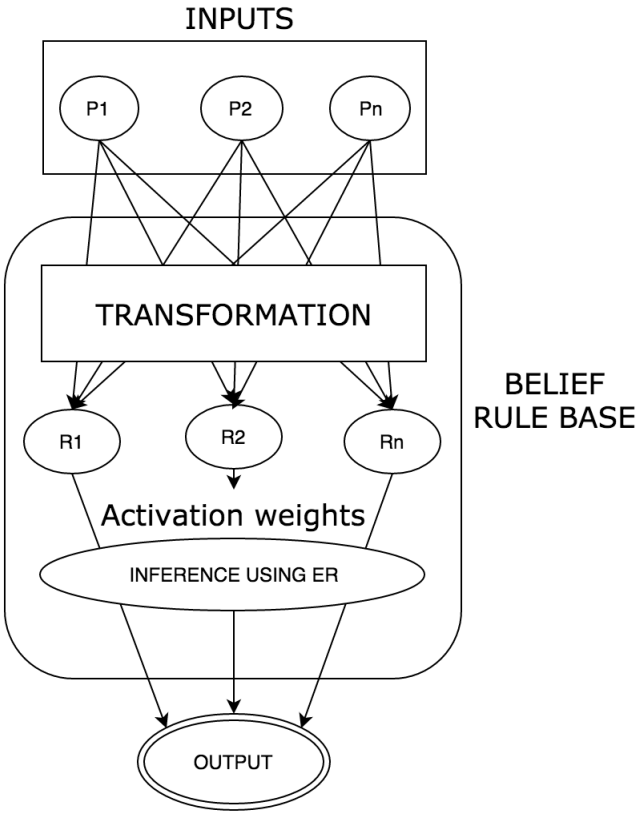

Figure 3: BRB inference architecture

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 System architecture proposed for flood detection

Andersson, K et. al. [14] 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.

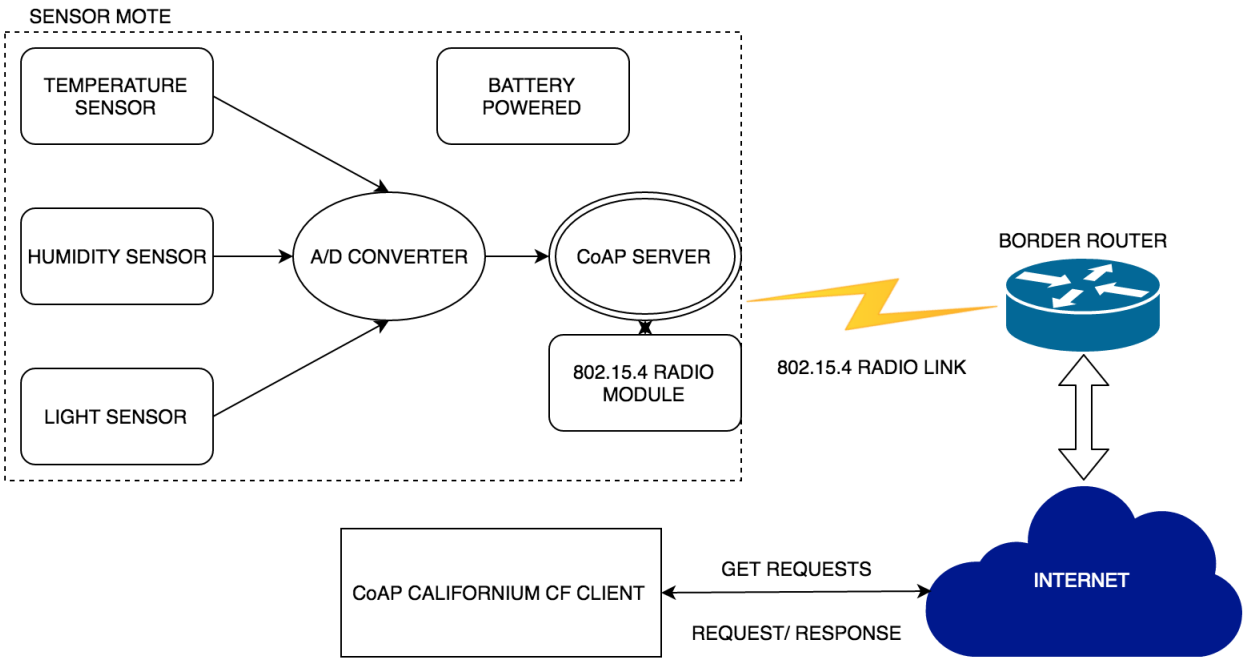

Figure 4: Proposed system architecture

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. 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 [15]. 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) [16] 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.

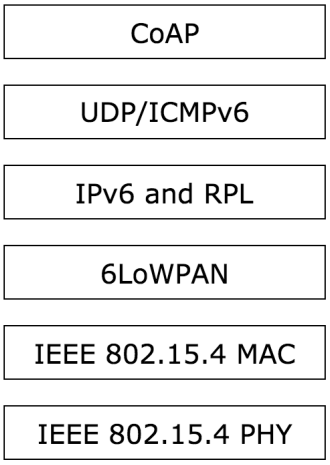

Figure 5: The WSN protocol stack

4 Tools and Technologies

This 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. 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.

4.1 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.

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) [17]. Contiki includes a very small implementation of IP called uIP [18], 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 6. 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.

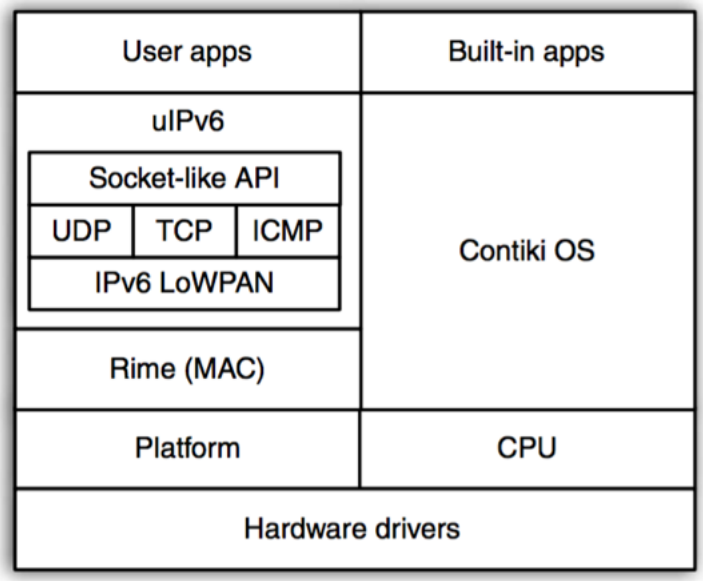

Figure 6: The Contiki architecture

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 work employs Contiki OS as the default OS, for both simulations and physical deployment.

4.2 Californium Cf

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 [19]. It is explicitly designed for scalable IoT cloud services and is publicly available at the Eclipse Foundation [20]. All these advantages motivated the use of this framework in the simulation scenarios and physical deployments for performance studies.

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 7: Californium code snapshots

4.3 Cooja

The various simulators researched for this work were OPNET, ns3, COOJA and OMNET++. 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.

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 [21].

5 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.

5.1 Simulation scenario

The simulation scenario [22] 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 8,

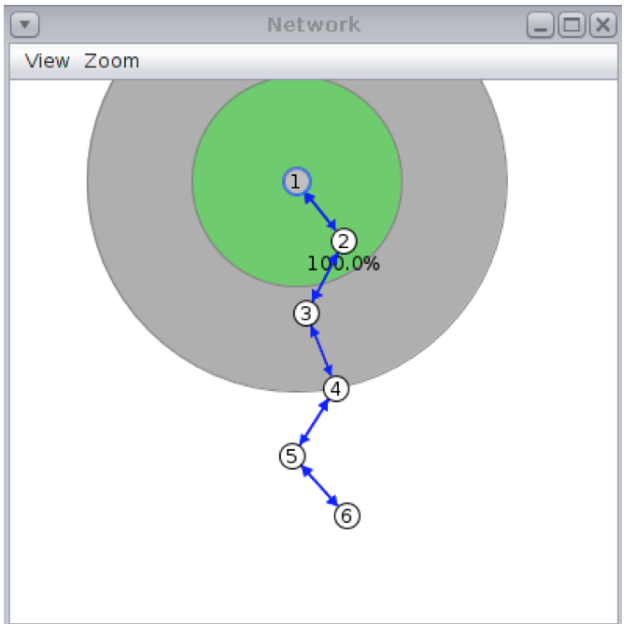

Figure 8: Simulation scenario network topology

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 1.

Table 1: General simulation parameters

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 2: Californium properties

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

5.2 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 directed acyclic graph (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.

5.2.1 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 3.

Table 3: Throughput variation (in Kbps) Vs Packet generation rate (in requests/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 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.

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.

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.

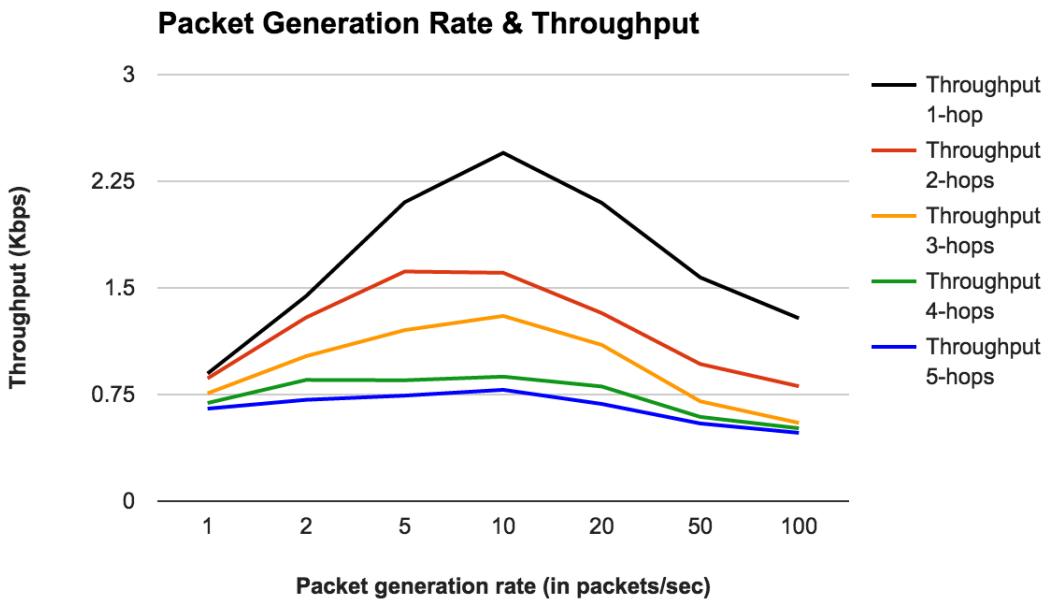

Figure 9: Average throughput for various motes at 1 to 5 hops against the rate of packet generation

5.2.2 End to end delay

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.

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 4: End to end delay variation (in ms) Vs packet generation rate (in req/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

It is the sum of the processing delay and twice the latency for the request and response to traverse through the network.

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.

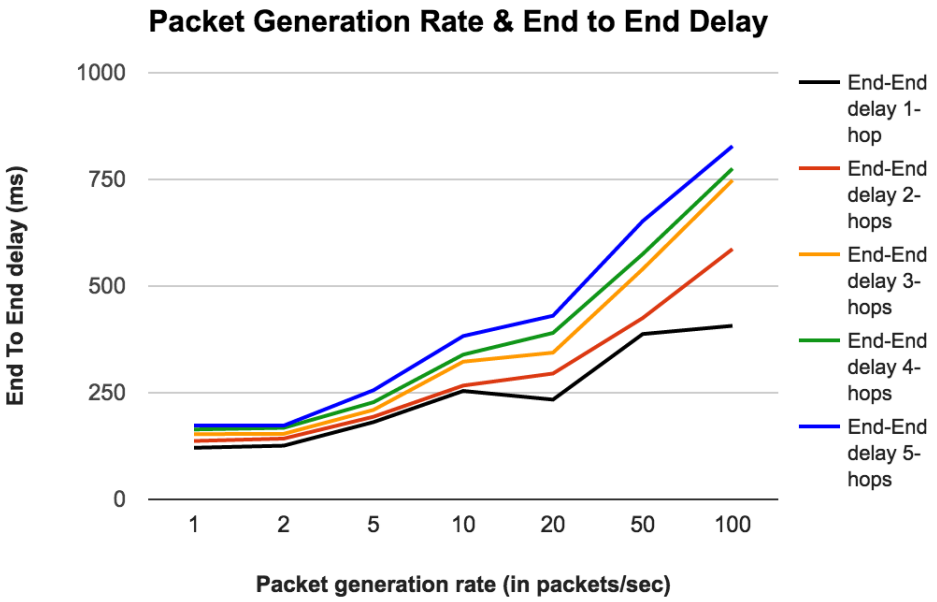

Figure 10: Plot of end to end delay for various motes at 1 to 5 hops Vs the rate of packet generation

5.2.3 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 values obtained are specified in table 5.

Table 5: Packet loss (in %) Vs packet generation rate PGR (in req/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 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.

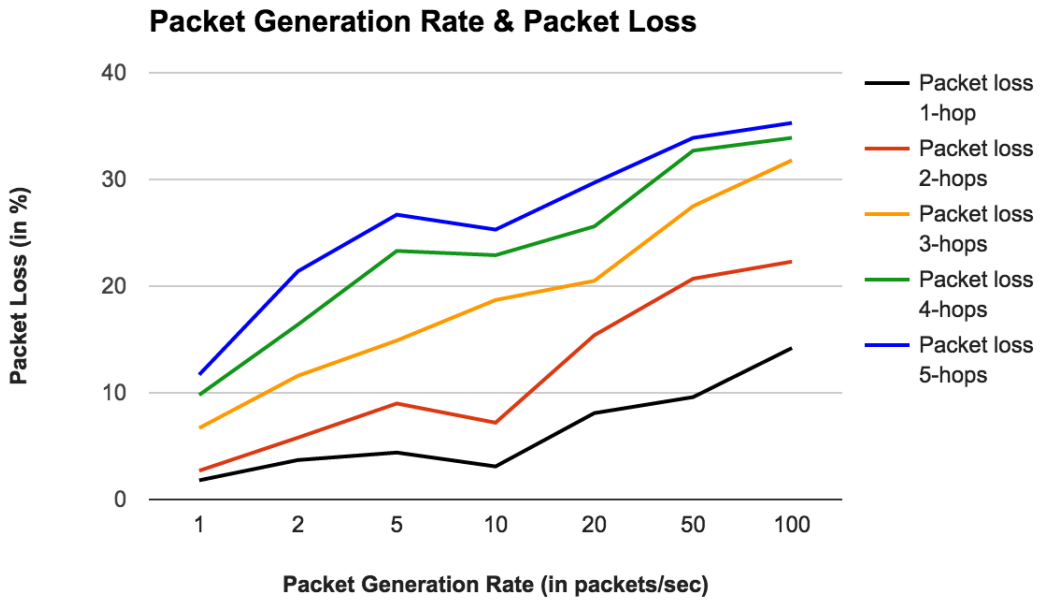

Figure 11: Plot of packet loss for various motes at 1 to 5 hops Vs the rate of packet generation

6 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. Due to this, the gateways can be small devices as this design puts the emphasis on end nodes exposing their resources running CoAP servers.

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. This work selected the hardware platforms, based on their compatibility with Contiki OS. The sky mote platform was found to be the most suitable for this work, due to its higher flash memory to run CoAP servers and relatively more documentation and availability of its modules. Within the sky platform there are further models 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.

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 work considered Raspberry Pi and Arduino Mega to function as border routers.

6.1 Deployment Scenario

Ease of deployment is an important metric when it comes to wireless sensor networks. The deployment scenario 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.

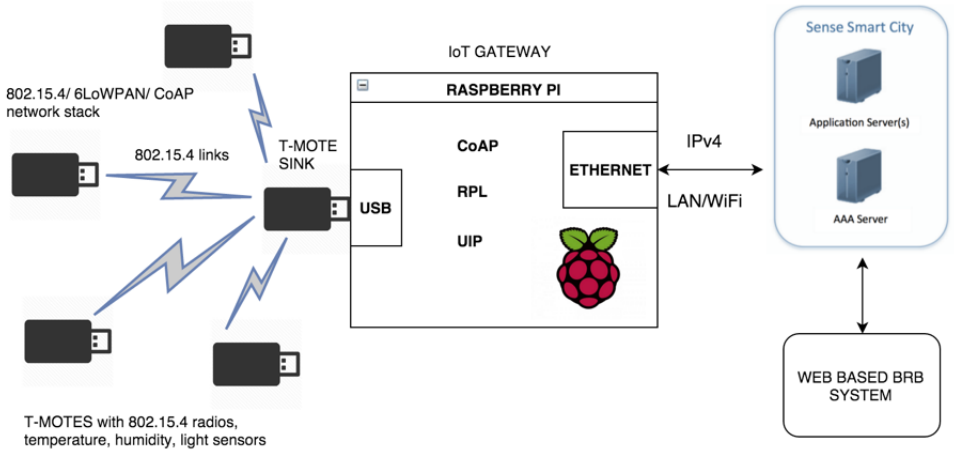

Figure 12: Scenario using RPi as a border router, sky motes running CoAP servers and a sky mote as a slip radio connecting the IP and the wireless domains

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. The hardware used is sky motes, running CoAP servers. This scenario is designed around the 6LBR project [23], 6LoWPAN border router solution. The ease of deployment is also high for 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.

6.2 6LBR

CETIC 6LBR [23] 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.

6.2.1 The router mode 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.

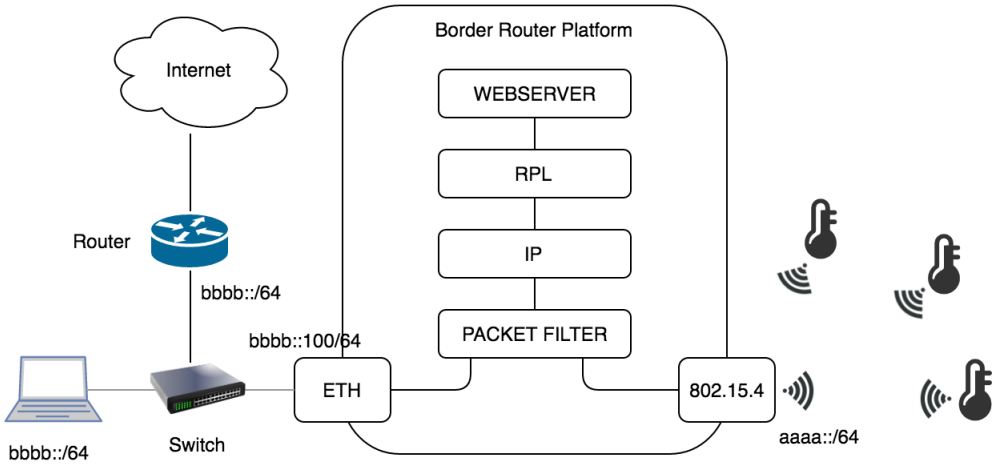

Figure 13: 6LBR - router mode

6.3 Experimental setup

Scalability analysis is an accurate measure of the reliability of the physical deployment. 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 14).

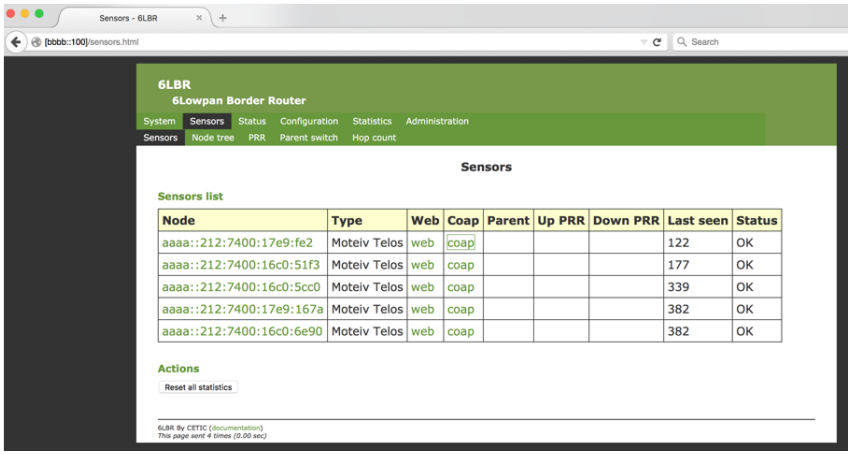

Figure 14: Webserver with various CoAP servers discovered in the area

Figure 15: RPi running 6LBR, connected over USB to the slip radio and via Ethernet to a development PC and the various sensors running CoAP servers

6.4 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 6: 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

6.4.1 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.

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

Table 7: 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

the slip radio and finally the WSN, giving more data to be transmitted.

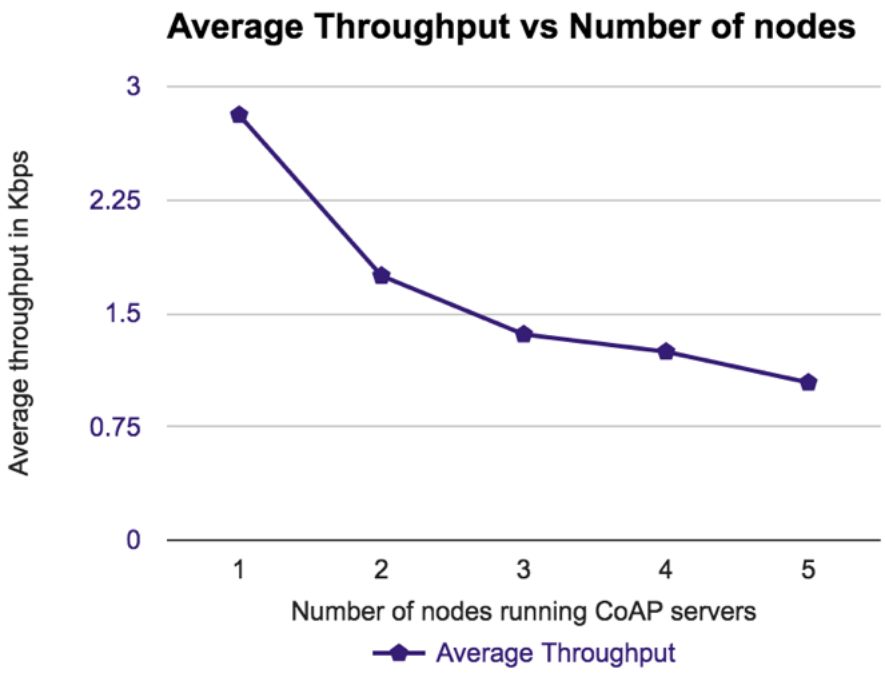

Figure 16: Plot of average throughput (in Kbps) as a function of no of motes running CoAP servers

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 greater than 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.

6.4.2 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 8: 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

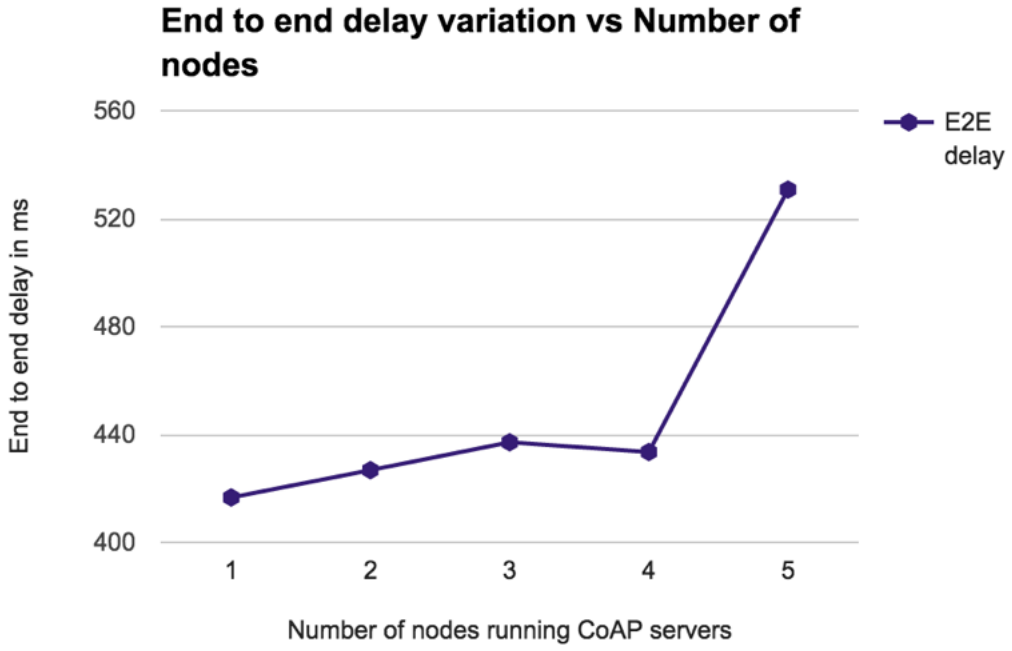

Figure 17: Plot of average end to end delay (in ms) as a function of no of motes running CoAP servers

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.

6.4.3 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,

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.

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 retransmis-

Table 9: 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

sion factor was changed to 4, according to the 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.

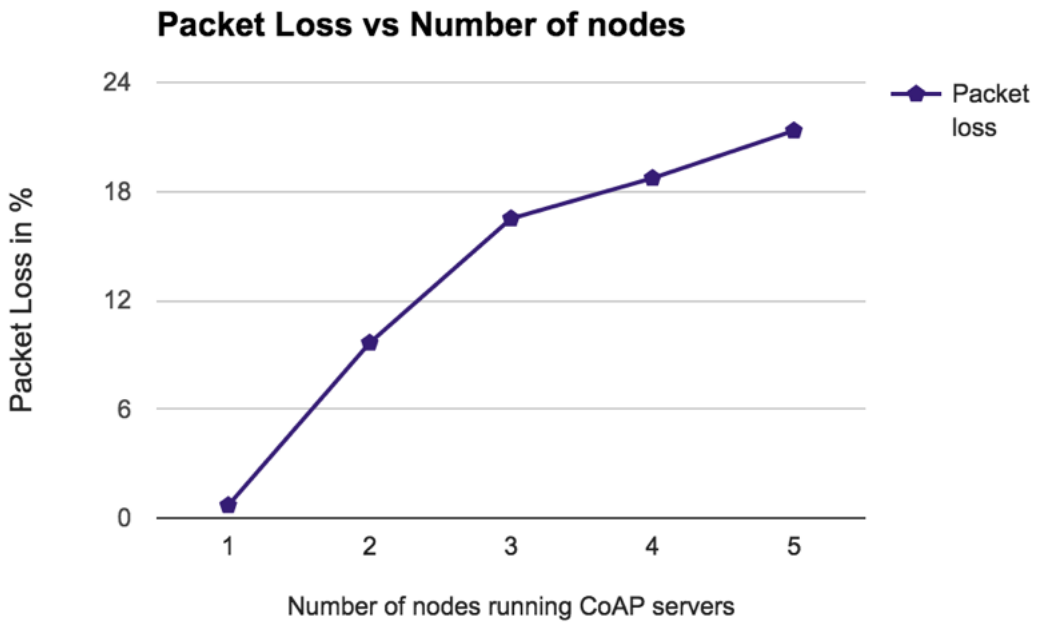

Figure 18: Plot of packet loss (in %) as a function of no of motes running CoAP servers

7 Conclusion

The work uses an IP based network stack and directly enables the development of smart city applications, like flood detection. The comparison of simulated and experimental stack indicates the values shown in table 10, in terms of 5 motes and a packet generation rate of 10ms.

Table 10: 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 metric of interoperability is the crucial aspect this work tries to address. The stack can be used with 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.

Reliability is an onus on the application layer protocol, CoAP in this case, since it is over an unreliable UDP at the transport layer. The results prove the reliability of the stack in terms of packet loss. The tuning of the CoAP retransmission factor allows this increased reliability by incorporating re-transmissions, particularly useful in noisy environments. When this factor was set to a value of 4, negligible packet loss was observed for both simulations and physical experiments, even when the number of motes was 5 and all the 1000 requests were sent all at once. This is a testimonial to the reliability prowess of the stack and is therefore very suitable for environmental applications like flood assessment, susceptible to noise and interference in the atmosphere.

The demonstrated stack 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. 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.

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.

The work also promotes the aspect of sustainability, by reducing vendor lock-in, using open standards and interoperable hardware components making application gateways redundant, thus reducing the amount of hardware manufactured. As a consequence of applying this stack to smart city applications such as flood detection in this work, it also benefits the triad of economy, ecology and equity.

8 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 [24], and various CoAP implementations differ in the usage of this timeout factor computation.

In the physical deployment, the 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, different 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.

Acknowledgments

This research was conducted as part of the master thesis of Sumeet Thombre, at the Pervasive and Mobile Computing Laboratory, Luleå University of Technology, Sweden. The research work is financially supported by the Swedish Research Council (under the grant 2014-4251) and the Erasmus Mundus program PERCCOM [25], facilitated by the European Commission.

References

[1] "IEEE, The Institute, Special Report: The Internet of Things," http://theinstitute.ieee.org/static/special-report-the-internet-of-things [Online; Accessed on September 10, 2016].

[2] "IEEE P2413 working group, Standard for an Architectural Framework for the Internet of Things," http://standards.ieee.org/develop/project/2413.html [Online; Accessed on September 10, 2016].

[3] G. M. Lee, J. Park, N. Kong, and N. Crespi, "The Internet of Things: Concept and Problem Statement," IETF Internet-draft (work in progress), July 2012, https://tools.ietf.org/id/draft-lee-iot-problem-statement-05.txt.

[4] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 packets over IEEE 802.15.4 Networks," IETF RFC 4944, September 2007, http://www.ietf.org/rfc/rfc4944.

[5] G. Mulligan, "The 6LoWPAN Architecture," in *Proc. of the 4th workshop on Embedded Networked Sensors (EmNets'07), Cork, Ireland*. ACM, June 2007, pp. 78–82.

[6] "The ZigBee Alliance," December 2014, http://www.zigbee.org [Online; Accessed on September 10, 2016].

[7] J. Sarto, "ZigBee Vs 6LoWPAN for sensor networks," White Paper, LSR, https://www.lsr.com/white-papers/zigbee-vs-6lowpan-for-sensor-networks [Online; Accessed on September 10, 2016].

[8] "The Eclipse Newsletter, IoT Protocols, CoAP and MQTT," http://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php [Online; Accessed on September 10, 2016].

[9] F.-Y. Leu, H.-L. Chen, and J.-C. Liu, "Improving multi-path congestion control for event-driven wireless sensor networks by using TDMA," *Journal of Internet Services and Information Security (JISIS)*, vol. 5, no. 4, pp. 1–19, November 2015.

[10] C.-E. Weng, V. Sharma, H.-C. Chen, and C.-H. Mao, "PEER: Proximity-Based Energy-Efficient Routing Algorithm for Wireless Sensor Networks," *Journal of Internet Services and Information Security (JISIS)*, vol. 6, no. 1, pp. 47–56, February 2016.

[11] R. Willows, N. Reynard, I. Meadowcroft, and R. Connell, "Climate adaptation: Risk, uncertainty and decision-making," UKCIP Technical Report, May 2003, http://www.ukcip.org.uk/wp-content/PDFs/UKCIP-Risk-framework.pdf [Online; Accessed on September 10, 2016].

[12] K. Andersson and M. S. Hossain, "Smart risk assessment systems using belief-rule-based DSS and WSN technologies," in *Proc. of the 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE'14), Aalborg, Denmark*. IEEE, May 2014. [Online]. Available: http://dx.doi.org/10.1109/VITAE.2014.6934397

[13] J.-B. Yang, J. Liu, J. Wang, H.-S. Sii, and H.-W. Wang, "Belief rule-base inference methodology using the evidential reasoning approach-RIMER," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 36, no. 2, pp. 266–285, March 2006.

[14] K. Andersson and M. S. Hossain, "Heterogeneous wireless sensor networks for flood prediction decision support systems," in *Proc. of the 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'15), Hong Kong, China*, April-May 2015, pp. 133–137.

[15] "The IPSO Alliance," http://www.ipso-alliance.org/about [Online; Accessed on September 10, 2016].

[16] G. Klas, F. Rodermund, Z. Shelby, S. Akhouri, and J. Höller, ""Lightweight M2M": Enabling Device Management and Applications for the Internet of Things," White Paper, February 2014, http://theinternetofthings.report/Resources/Whitepapers/c9b5e66c-ffb9-4011-8578-49bc01075085_1_28701-FGB101973_EN_B_PDFV1R1.pdf [Online; Accessed on September 10, 2016].

[17] A. Dunkels, O. Schmidt, N. Finne, J. Eriksson, F. Österlind, and N. Durvy, "The Contiki OS: The Operating System for the Internet of Things," http://www.contiki-os.org [Online; Accessed on September 10, 2016].

[18] A. Dunkels, "Full TCP/IP for 8-bit architectures," in *Proc. of the 1st international conference on Mobile Systems, Applications and Services (MobiSys'03), San Francisco, California, USA*. ACM, May 2003, pp. 85–98.

[19] M. Kovatsch, M. Lanter, and Z. Shelby, "Californium: Scalable cloud services for the internet of things with CoAP," in *Proc. of the International Conference on the Internet of Things (IOT'14), MIT Media LAB, Cambridge, Massachusetts, USA*. IEEE, October 2014, pp. 1–6. [Online]. Available: http://dx.doi.org/10.1109/IOT.2014.7030106

[20] "Californium: A CoAP Framework in Java," http://www.eclipse.org/californium [Online; Accessed on September 10, 2016].

[21] L. B. Saad, C. Chauvenet, and B. Tourancheau, "Simulation of the RPL Routing Protocol for IPv6 Sensor Networks: two cases studies," in *Proc. of the 2011 International Conference on Sensor Technologies and Applications (SENSORCOMM'11), Nice, France*. IARIA, September 2011.

[22] S. Thombre, R. U. Islam, K. Andersson, and M. S. Hossain, "Performance Analysis of an IP based Protocol Stack for WSNs," in *Proc. of the 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'16), San Francisco, California, USA*. IEEE, April 2016, pp. 691–696.

[23] "CETIC 6LBR: a 6LoWPAN/RPL Border Router," https://github.com/cetic/6lbr/wiki [Online; Accessed on

September 10, 2016].

[24] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," IETF RFC 7252, June 2014, http://www.ietf.org/rfc/rfc7252.txt.

[25] A. Klimova, E. Rondeau, K. Andersson, J. Porras, A. Rybin, and A. Zaslavsky, "An international Master's program in green ICT as a contribution to sustainable development," *Journal of Cleaner Production*, vol. 135, pp. 223–239, November 2016.

Author Biography

Sumeet Thombre received his B.E. degree in Electrical Engineering from the Visvesvaraya Technological University, India in 2011. He is currently pursuing his Masters from the Luleå University of Technology, Sweden as part of an Erasmus Mundus Masters degree in PERCCOM. He has previously been associated with Robert Bosch, as a senior software engineer in the field of automotive embedded electronics. His research interests include Wireless Sensor Networks, the Internet of Things, Embedded Systems, and Automotive Electronics.

Raihan Ul Islam is a Ph.D. candidate at the Luleå University of Technology, Sweden. His main research focus is on Expert Systems, Machine Learning and Smart Cities. He received his M.Sc. degree in Computer Science from Luleå University of Technology, Luleå, Sweden. Previously he worked as a software engineer at NEC Laboratories Europe, in the Context-aware Services (CAS) and Smart Environments Technologies Group. His research interests also include Machine Learning, M2M Communication, Smart Homes and Cities, Mobile Systems, and Pervasive and Ubiquitous Computing.

Karl Andersson has a M.Sc. degree in Computer Science and Technology from Royal Institute of Technology, Stockholm, Sweden and a Ph.D. degree in Mobile Systems from at Luleå University of Technology, Sweden. After being a postdoctoral research fellow at the Internet Real-time Laboratory at Columbia University, New York, USA and a JSPS Fellow with National Institute of Information and Communications Technology, Tokyo, Japan, he is now Associate Professor of Pervasive and Mobile Computing at Luleå University of Technology, Sweden. His research interests include Mobile Computing, the Internet of Things, Cloud Technologies, and Information Security. He is a Senior Member of IEEE.

Mohammad Shahadat Hossain is a Professor of Computer Science and Engineering at the Chittagong University (CU), Bangladesh. He did both his MPhil and PhD in Computation from the University of Manchester Institute of Science and Technology (UMIST), UK in 1999 and 2002 respectively. His current research areas include e-government, the modeling of risks and uncertainties using evolutionary computing techniques. Investigation of pragmatic software development tools and methods, for information systems in general and for expert systems in particular are also his areas of research.