ABSTRACT

SECURE SYSTEM SIMULATION - INTERNET OF THINGS

By

Yukti Verma

May 2016

Internet of Things (IoT) can be defned as a collection of smart devices interacting with each other unanimously to fulfill a common goal. The real world data collected from the Internet of Things can be made as an integral part of web known as Web of Things(WoT). With the help of Web of Things architecture, the users can leverage simple web mechanisms such as browsing, searching and caching to interact with the smart devices. This thesis aims to create an entire system simulating the Web of Things architecture including sensors, edge routers, web interfaces, endpoints to the IoT network and access control. Several technologies such as CoAP, 6LoWPAN, IEEE 802.15.4, contiki and DTLS have been evaluated before inclusion in the implementation. A complete web portal utilizing Californium framework and Role Based Access Control has been created for accessing and interacting with the sensors and their data. This thesis provides an end-to-end approach towards IoT device security by implementing Constrained Application Protocol(CoAP) over Datagram Transport Layer Security(DTLS) in the system. The performance of secured system is analyzed in a constrained environment based on which it is observed that DLTS implementation increases the RAM usage, code size, packet overhead and power consumption by a significant value. Finally, the future work that needs to considered in order to iterate towards better security is specified.

SECURE SYSTEM SIMULATION - INTERNET OF THINGS

A THESIS

Presented to the Department of Computer Engineering and Computer Science

California State University, Long Beach

In Partial Fulfilment

of the Requirements for the Degree

Master of Science in Computer Science

Committee Members:

Mehrdad Aliasgari, Ph.D. (Chair)
Mohammad Mozumdar, Ph.D.
Burkhard Englert, Ph.D.

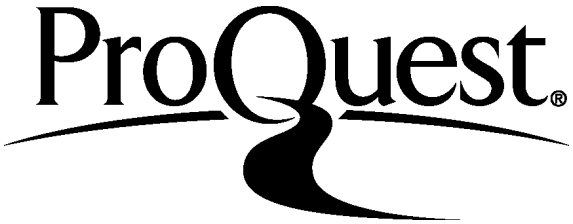
College Designee:

Antonella Sciortino, Ph.D.

By Yukti Verma

B.Tech., Computer Science, 2013, Maharishi Dayanand University, India

May 2016



ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my advisor Dr. Mehrdad Aliasgari for his continuous support throughout my journey of pursuing my thesis under his guidance. He has been an encouraging and supporting mentor who has always given a fleexible environment for me to work and explore my abilities. His immense knowledge was an asset that was always readily available for me to explore, which eventually turned out to be the best source of information.

Thank you to the Library and the Thesis and Dissertation Office at California State University, Long Beach for the availability of resources for my research work and for the detailed guidelines available online for writing the thesis report.

Last but not the least, thanks to all my family members and friends for their constant support and love throughout my research work.

TABLE OF CONTENTS

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER 1

INTRODUCTION

Internet of things(IoT) is a concept of connecting the everyday objects to the internet. The physical objects can be as small as heart monitoring implants, bio-chip transponders on farm animals, phones, coffee makers, lamps or as big as machines such as jet engines, automobiles, oil rig drills etc. The IoT savvy experts estimate that by the end of 2020, at least 26 billion objects will be connected to the internet. This may seem like a bold statement that anything that can be connected will be connected. Undoubtedly, IoT represents the next evolution and will become one of the most powerful tools in the human history resulting in improved efficiency, accuracy and economic benefits.

Purpose and Goals

The purpose of this master's thesis is to create Web of Things architecture with end to end IoT security and examine several aspects of implementation including size, memory usage and power consumption in a constrained environment. To examine the aspects, many technologies and protocols need to be analyzed, and then a decision about what to use has to be made based on a set of requirements. We plan to create an end-to-end approach towards IoT device security by implementing Constrained Application Protocol(CoAP) over Datagram Transport Layer Security(DTLS) in the system and then analyze the performance of the created system in a constrained environment. Also, a complete web portal with Role Based Access Control(RBAC) needs to be created for accessing and interacting with the sensors and data using Californium framework.

Background

Protocols and Networks

IEEE 802.15.4. IEEE 802.15 concentrates on the standardization of the Wireless Personal Area Networks and also categorizes Wireless Sensor Networks (low powered) in the IEEE 802.15.4 standard [1].The IEEE 802.15.4 standard is also known as Low Rate-Wireless Personal Area Network(LR-WPAN). There is a set of recommended features that a link should provide in IoT. They are easy to install and provide reliable data transfer. As they use unlicensed radio bands, they are extremely low cost, flexible and extendable networks [1]. This standard defines two layers of Low Rate Wireless Power Area network(LR-WPAN). The physical layer consists of 27 channels divided into three different bands of 2450 MHz, 915 MHz and 868 MHz frequency respectively. Activation/Deactivation of radio transceiver, Channel Selection, Energy Detection are some of the tasks performed by this layer. The Media Access Control(MAC)-Sub layer acts as an interface between the service specific convergence sub layer and the physical layer. Generating and managing beacons, channel access, time management and frame validation are some of the tasks performed by the MAC layer.

Bluetooth Low Energy(BLE). BLE technology is intended to provide considerably reduced power consumption and cost while maintaining a similar communication range [2]. They have low data rate and the protocol is optimized to burst transmit small blocks of data at regular intervals, thus enabling the host processor to maximize the amount of time it can operate in a low power mode when information is not being transmitted. Each layer of the architecture has been optimized to reduce power consumption [3]. The modulation index of physical layer is increased which helps in reducing transmit. The link layer does quick reconnects in order to

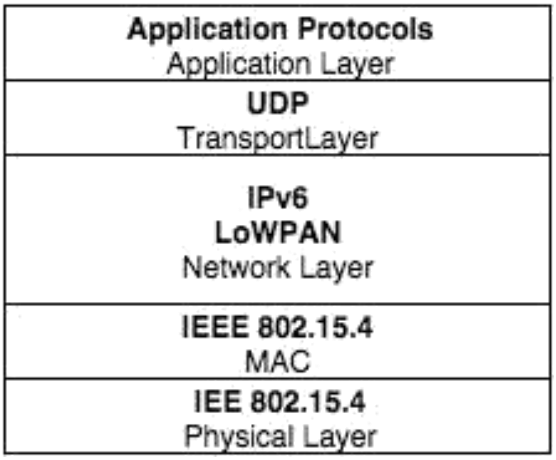


FIGURE 1. 6LoWPAN layers.

reduce the power. The controller helps the host processor to stay in low power mode for an extended duration by ignoring the duplicate packets. The network topology of BLE consists of single master and one or more slave nodes. The slave nodes only wake up periodically to listen for the data in order to save energy. BLE does not only specify the physical and link layers, but also higher layers. Generic ATTribute pro le (GATT) is an application layer protocol that allows exchange of data in the form of properties between devices. GATT does only exist in BLE [4].

6LoWPAN. 6LoWPAN (IPv6 over Low-rate WPAN) working group was formally established by IETF to institute LR-WPAN standard based on IPv6. The purpose of this group is to introduce IPv6 into LR-WPAN which takes IEEE 802.15. 4 as its basic bottom layer standard [5]. The working group constructs self organization 6LoWPAN network with the route protocol. The physical and MAC layer, as shown in Figure 1, of 6LoWPAN technology adopts IEEE 802.15.4 standard. Some of the advantages of 6LoWPAN are more address space available, stateless address auto configuration support. Multi hopping, flexibility, fragmentation and reassembly are some of the important features of 6LoWPAN technology. It establishes three main types

FIGURE 2. CoAP abstract view.

of architectures based on different types of LoWPAN, i.e., ad-hoc LoWPAN, simple LowPAN and extended LowPAN.

CoAP. The Constrained Application Protocol is a web transfer protocol specially designed for the constrained nodes and networks. It is a request/response interaction model which helps in integration with existing web along with satisfying special needs of constrained devices. It is similar to WWW-HTTP. Along with low overhead, this protocol supports asynchronous messages and caching possibilities. The abstract layering of the CoAP protocol is shown in Figure 2. It is based on REST model in which the resources are available under a URL and clients access these resources using GET, PUT, POST and DELETE methods. It also provides support for discovery of resources.

Datagram Transport Layer Security(DTLS). Datagram Transport Layer Security(DTLS) is a derivation of SSL protocol except that it is designed for User Datagram Protocol(UDP) instead of Transmission Control Protocol(TCP). The protocol elements of TLS are more or less similar to the DTLS [6].

Reliable, in-order packet delivery, replay detection are some of the features that are absent from the DTLS. The experiments show that DTLS handshake message fragment has more than double the overhead from headers compared to TLS [6] The DTLS can be enabled with CoAP using Pre Shared Key, raw public key, or certificates.

MQTT. A machine to machine connectivity protocol as shown in Figure

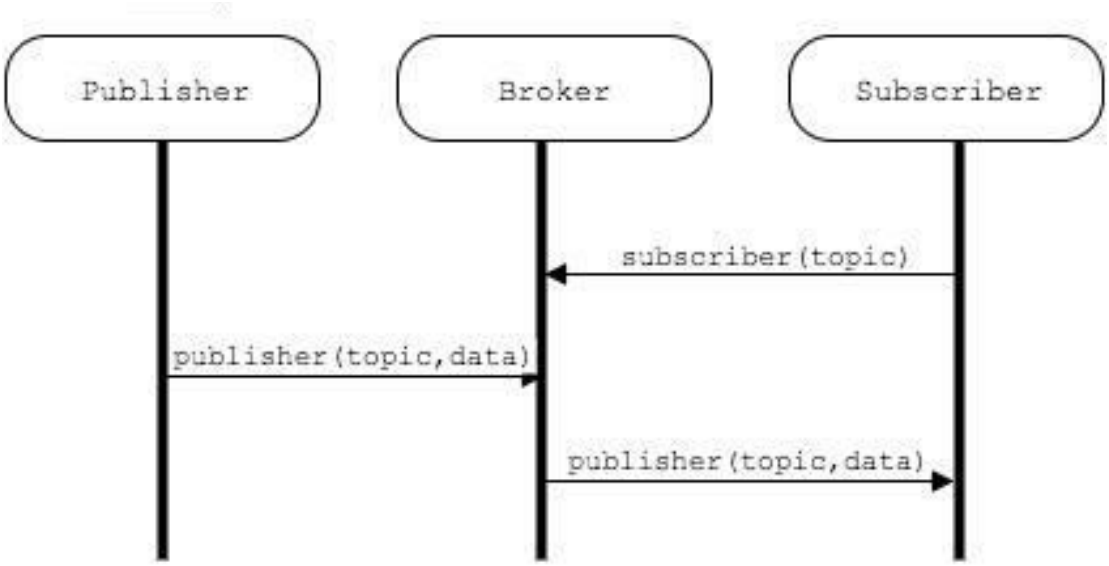


FIGURE 3. MQTT protocol.

3 is designed as a publish/subscribe messaging transferring service where each sensor is a client and connects to the server over TCP. Message Queue Telemetry Transport for Sensor networks(MQTT-S) is designed in such a way that it can be run on low-end and battery-operated sensor/actuator devices and operate over bandwidth-constraint Wireless Sensor Networks(WSNs) such as ZigBee-based networks[7].

Hardware

CC2650 Sensor Tag. CC2650 Sensor Tag is a low power, cost effective 2.4 GHz wireless MCU with more than 10 sensors including light, humidity, pressure.

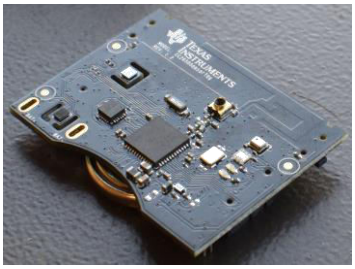


FIGURE 4. TI CC2650.

FIGURE 5. TI CC2531.

digital microphone embedded in it [8]. This small piece of hardware, shown in Figure 4, can act as a mesh device that interacts with the edge router for transferring the data. It consists of ARM Cortex M3 processor with 128 KB of ash memory. It also adds support for more low power sensors. One of the most important features is that they can be enabled on Zigbee or 6LoWPAN. The debugger dev pack or SmartRF06 evaluation board is required to add the debug capability to the sensor tag.

TI CC2531. This piece of hardware shown in Figure 5 is a USB enabled true system on chip solution for IEEE 802.15.4 applications [9]. A 2.4 GHz IEEE 802.15.4 Compliant RF Transceiver has excellent receiver sensitivity to interference. It can be used as an 6LoWPAN edge router containing a serial line IP application, translating between RF and serial line IP. Prior to using this as a 6LoWPAN edge router, it needs to be flashed with the slip radio application provided. The USB dongle requires CC debugger with Smart RF flash programmer for flashing the memory

BeagleBone Black(BBB). The BeagleBone Black board is a low-power community supported hardware single- board computer development platform for developers produced by Texas Instruments in association with Digi-Key and Newark element14 [2]. It contains 512 MB RAM, 4GB flash storage and two PRU 32 bit micro controllers with USB, Ethernet and HDMI connectivity.

Software

Contiki. Contiki is an open source operating system specially designed for memory constrained low power wireless Internet of Things devices [10]. Contiki supports powerful low-power Internet communication and fully standard IPv6 and IPv4 [11]. Some of the features of Contiki OS are memory efficiency-specially designed for memory constrained tiny systems, therefore this OS provides mmem, managed memory allocator and malloc, the standard C memory allocator. 6LoWPAN, CoAP support- contiki supports the recent low power wireless standards including the 6LoWPAN adaption layer and CoAP RESTful protocol. Linking of modules at runtime- this feature is useful if the behavior of the application is likely to change after the deployment. Contiki does not provide any security features. There are some external security libraries that can be used with contiki. The CC2531 has hardware accelerated AES-128 encryption and decryption, which is suitable for security functionality.

COOJA. COOJA is a network simulator provided by the contiki OS to ease the process of developing and debugging software of large wireless networks. Also, Contiki can be run on large variety of platforms like ARM devices, MSP430 etc.

Instant contiki. The instant contiki does not need any complex compilers so it can be run on a simple Linux machine.

ARM MBeD. Mbed is another operating system for constrained devices based on ARM Cortex- M. It supports IPv6, 6LoWPAN adaption layer

RPL routing protocol, IPv4 stack, Mesh link establishment. It can be run as a bare-metal stack as well as with ARM Mbed OS, an open source embedded operating system designed specifically for the things in IoT [12]. It includes security modules like DTLS, TLS, PANA, PSK. 6LBR It is a 6LoWPAN deployment ready border router solution from CETIC based on contiki. It does not require a Linux host for routing while smartly interconnecting IPv6 mechanisms and RPL [13]. 6LBR helps WSN network, based on 802.15.4, 6LoWPAN and Ethernet based IPv6 network to communicate with each other. It has three different modes of operation: router, bridge and transparent bridge. The connection can be made at different layers of network stack like at link layer the bridge category mode can be used, at network layer the router category mode can be used . The router mode is further divided into four modes which are routers, ndp-router, 6LR, RPL-Root and the transparent bridge is divided into two modes which are RPL-Replay, Full Transparent Bridge.

6LBR. It is a 6LoWPAN deployment ready border router solution from CETIC based on contiki. It does not require a Linux host for routing while smartly interconnecting IPV6 mechanisms and RPL [13]. 6LBR helps WSN network, based on 802.15.4, 6LoWPAN and Ethernet based IPv6 network to communicate with each other as shown in figure 6. It has three different modes of operation: router, bridge and transparent bridge. The connection can be made at different layers of network stack like at link layer the bridge category mode can be used, at network layer the router category mode can be used . The router mode, shown in Figure 6, can further divided into four modes: routers, ndp-router, 6LR, RPL-Root and the transparent bridge is divided into two modes RPL-Replay, Full Transparent Bridge.

Frameworks

Californium. Californium is a java based CoAP framework for back-end services and stronger Internet of Things devices [14]. It provides a convenient API for

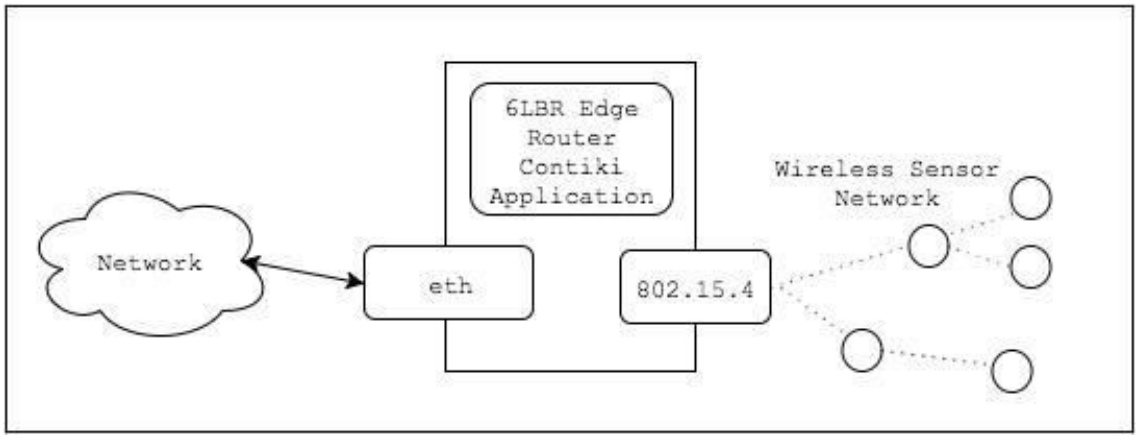


FIGURE 6. 6LBR edge router configuration.

RESTful web services that support all of the CoAP's features [15]. The Californium module is divided into five sub modules: Californium core that provides protocol implementation for Internet of Things, scandium for security, actinium supports javascript apps, CoAP tools for some examples and connectors that provide the basic connectors like UDPconnector. Copper tool-an add-on for the firefox can be used to browse, bookmark the CoAP resources.

Libcoap. Libcoap is a C based lightweight application-protocol implementation for constrained devices. This protocol was standardized in the IETF as RFC 7252. It is designed to run on embedded devices as well as high end computer systems with POSIX OS [16]. It is possible to develop the application on the laptop, test it and then move to the required platform. The core functionalities required for CoAP servers and clients are frameworks such as open SSL or tinydtls.

CHAPTER 2

RELATED WORK

Karlsson et al. [4] examine the possible limitations that can be imposed on the functionality of resource constrained wireless devices. The authors evaluate many IoT related technologies and their performance aspects. In the eld of wireless technologies, they looked into three different options: IEEE 802.15.4, Bluetooth Low Energy and WiFi. They conclude that IEEE 802.15.4 and BLE are both suitable to be used as wireless technologies as their maximum payload is high. They considered RESTful services and SOAP services but because SOAP requires more overhead, compared to RESTful design, RESTful services were used. After comparison they used the following technologies for implementation: IEEE 802.15.4 on the physical layer, 6LoWPAN on the network layer, CoAP over UDP on the application. The transport layer CoAP works on RESTful services and can translate HTTP into more compact, binary format. They conclude that CoAP server and the energy saving features cannot fit at same time. Even if we fit them together, the energy saving protocol is not efficient to use small button batteries. CoAP together with 6LoWPAN seems to work well. The contiki OS was used to simplify the implementation as it provides support for most of the protocols required on every layer. However, no security parameters were considered while assessing these limitations.

Ramrezz [17] has developed and tested a border router solution. The author has analyzed the performance of open labs 802.14.5 module with the 6LBR border router and concluded that the combination was not successful in terms of connectivity.

Also, he compared a lot of border router solutions such as redwire BR12, cisco, nanopower, openlabs and 6LBR in terms of RPi compatibility, cost, open WSN concluding that 6LBR is a very interesting solution offering a considerable variety of operation modes and powerful web administration to get a starting point to develop web services and OAM applications. He also considered the environmental impacts of the project taking into account power consumption, reusability, reduced use of materials.

Kovatsch et al. [14] propose a system architecture for scalable IoT cloud services based on the Constrained Application Protocol (CoAP), which is primarily designed for systems of tiny, low-cost, resource-constrained IoT devices. Their work is inspired by Staged Event-Driven Architecture(SEDA) [18]. SEDA divides the message handling process into multiple stages similar to what authors have proposed. They proposed a three stage architecture which includes network stage, protocol stage and business logic stage. The network stage is responsible for receiving and sending the byte arrays over the network. The protocol stage executes the CoAP protocol and has a thread pool and the business logic stage is role dependent(server and client). They also evaluate the performance of their new protocol and show that Californium (Cf) CoAP framework shows 33 to 64 times higher throughput than high-performance HTTP Web servers, which are the state of the art for classic cloud services. According to them the IoT cloud services and Web integration platforms need to speak CoAP directly to be able to scale to vast numbers of concurrently connected devices.

CHAPTER 3

CHOOSING TECHNOLOGIES

As we need to implement different efficient technologies at different layers, it is very important to compare the described technologies with one another and choose whichever suits the best. Precisely, we need to choose among one of the wireless technologies, a technology for the network layer, a transport layer security mechanism and an application layer protocol.

Assumptions

Limited resources: It is assumed that the sensor nodes have limited resources such as memory, power supply.

Consideration of current standards: Current standards such as IEEE 802.15.4 need to be considered and should be preferred.

Discussion

The options available for the wireless technologies are 6LoWPAN, BLE and WiFi. WiFi is a widespread wireless technology but not suitable for constrained devices because of high power consumption. The low power consuming technologies are IEEE 802.15.4 and the BLE. Although, the BLE is robust as it uses adaptive frequency hopping than single channel, but our nodes are based on mesh topologies and the drawback is that BLE does not support mesh topologies. Also, The maximum payload of IEEE 802.15.4 is higher than that of BLE . We can use ipv6 or 6LoWPAN with the wireless technologies, but IPv6's header is very large as compared to 6LoWPAN's which compresses the header and let messages have bigger size. We can use 6LoWPAN

with BLE or IEEE 802.15.4 but BLE does not have any written implementation of 6LoWPAN, therefore we would prefer IEEE 802.15.4 with 6LoWPAN. At the application layer, we have two options HTTP or CoAP. HTTP requires reliable transport protocol like TCP which would be an overhead for the constrained devices whereas the CoAP can run over UDP, a non reliable protocol, thereby reducing the overhead. Hence we chose CoAP over UDP for the Application layer.

In nutshell, we have chosen IEEE 802.15.4 on Physical Layer, 6LoWPAN on Network Layer, CoAP on Application Layer. As the data transfer between the gateway and the internet should be HTTP, we are required to translate the CoAP to HTTP. A lot of libraries and frameworks have been described in the introduction section for the CoAP to HTTP conversion, but we are going to use the Californium framework for translation as it provides more support for the security.

CHAPTER 4

IMPLEMENTATION

This section focuses on detailed explanation of the development process of the complete system

Setting up Gateway/Edge Router

To set up the edge router, a TI CC2531 USB dongle connected to a Linux system running a deployment ready 6LBR solution on it is used. A serial line IO application is required on the USB dongle in order to translate between a serial line and RF. The procedure that was used to set up the edge router is described below: Before using the CC2531 USB dongle as 6LoWPAN device, we need to download the cc2531 slip radio contikimac.zip file from the internet and flash the CC2531 USB dongle with this file. A CC Debugger with a Smart RF Flash Programmer is required for this process. After that 6LBR needs to be downloaded and configured on the Linux system by using the commands shown in Figure 7 [19]. Connect the USB dongle to the Linux system on which 6LBR was installed and start the edge router by using the commands shown in Figure 8.

Setting up Wireless Sensor Network

We have used a CC2650 sensortag which has more than 10 sensors embedded in it. This sensortag acts as a 6LoWPAN router and sends data to the server using the edge router. The CC2650 sensortag originally uses the BLE technology and can be accessed using the mobile app that is available for both android and IOS. To make it as a devpack debugger and flash programmer 2 available only for windows. The cc26xx-

```
git clone –recursive https://github.com/cetic/6lbr

cd 6lbr/examples/6lbr

make all plugins tools

sudo make install
```

A 6lbr configuration file `"/etc/6lbr/6lbr.conf"` needs to be created with the following content:

```
MODE=ROUTER
RAW_ETH=1
BRIDGE=0
DEV_BRIDGE=br0
DEV_TAP=tap0
DEV_ETH=eth0
RAW_ETH_FCS=0
DEV_RADIO=/dev/ttyACM0
BAUDRATE=115200
LOG_LEVEL=3
```

We need to change the channel to 25 which is used by cc2650 contiki using the following command.

`"/usr/lib/6lbr/bin/nvm_tool--update --channel 25 /etc/6lbr/nvm.dat"`

The settings can be checked using the nvm_tool. If the network prefix is not present in the settings then it can be manually added using:

```
route -A inet6 addaaaa::/64 gw bbbb::100
```

or RA processing can be allowed using the following two commands.

```
sysctl -w net.ipv6.conf.br0.accept_ra=1

sysctl -w net.ipv6.conf.br0.accept_ra_rt_info_max_plen=64
```

FIGURE 7. Edge router commands.

```
sudo service 6lbr start
and try to ping the eth segment using
ping6 aaaa::<address of mesh node>
```

FIGURE 8. Start 6LBR commands.

web-demo, provided by contiki [19], was configured and modified to use CoAP protocol and different sensors. The modified program also includes the client for 6LBR that can display the network topology in 6LBR webserver and a CoAP server that can be used to access the resources using the CoAP protocol. We also added the DTLS module inside the program for security.

In order to ash the program on the sensortag device, the program le was converted to bin file using the command shown in Figure 9.

```
make TARGET=srf06-cc26xx BOARD=sensortag/cc2650 cc26xx-web-demo.bin CPU_FAMILY=cc26xx
```

FIGURE 9. CC2650 setup commands.

The bin file was then transferred from Linux to windows for flashing purpose. To access the sensortag's resources from CoAP server, we first installed a Copper plugin on the Firefox web browser and then started the CoAP server from 6LBR webserver and accessed our resources via CoAP.

DTLS Implementation

The tinyDTLS library was chosen for the implementation of DTLS due to presence of PSK enabled *TLS_PSK_AES_128_CCM_8* and ECC enabled

TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 cipher suites. In order to integrate DTLS with CoAP, we had to make changes in er-coap(erbium-CoAP) application. er-coap is introduced by 6LBR CoAP in order to choose other transport protocols than UDP.The changes that were made in the files of erbium are listed in [20]. We implemented PSK enabled *TLS_ PSK_ AES_ 128_ CCM_ 8* suite in our application and analyzed its performance.

Due to memory restriction on the sensortag, we were not able to implement the *TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8* cipher suite in our application, so we used COOJA simulator to implement the ECC cipher suite and measure its performance on a wismote with CoAP server. We used 1-BR COOJA instance, provided by 6LBR, to generate a COOJA CSC file and compile COOJA. Serial2pty and Radiologger headless modules were added to the instance after the compilation. The steps to create the instance with 6LBR are mentioned in Figure 10.

To start the border router for COOJA, the steps mentioned in Figure 12 need to be followed which should open a screen as shown in Figure 11.

DTLS Client

The DTLS at the client side was implemented using Scandium, an open source DTLS implementation provided by Californium.

Server with Access Control

A server to access, control and monitor the sensors using the system structure shown in the gure was created with Role Based Access Control functionality embedded in it. Considering a smart home network,the system structure shown in Figure 13 can be used to control energy, operate electronics, get electronics' data or monitor the power consumption. The nodes of wireless sensor network could exchange information within themselves using the CoAP. The created control server provides CoAP-HTTP proxies

cd examples/6lbr/testmake cooja-small
This will do the following things

generate a COOJA .csc file
compile COOJA
compile COOJA plugins
launch COOJA with preset nodes

In order to implement DTLS, we added two more motes. The first mote is compiled with 6LBR-demo, TinyDTLS with ECC mode disabled, CoAP Server and DTLS-Echo server on it. The second mote has the same configuration but ECC mode enabled. Both of these new motes include powertrace app in order to keep track of energy.

The simulation now has five types of motes:
The green mote: It has a slip-radio application running on it in order to communicate with the edge router.
The pink motes: They are simple 6lbr demo motes which can run RPL on them.
The yellow mote(node id 12):This is a 6lbr demo mote which includes TinyDTLS ECC disabled, DTLS-Echo and CoAP Server.
The yellow mote(node id 13):This is a 6lbr demo mote which includes TinyDTLS ECC enabled, DTLS-Echo and CoAP Server.

FIGURE 10. 6LBR COOJA setup.

for HTTP client connection to CoAP resources and vice versa. This server is written in java and uses the Californium framework. The wireless network collects the data and sends serial data to proxy to process and pack data. Control server analyzes all the data and stores them in database. The clients can access the system via webpages to remotely control, manage or access the sensors.

Access Control

Access Control is necessary in the system as the system structure implies that devices will be available to the world to access. Therefore, device security will be at stake without proper access control. We have implemented the Role based access control(RBAC) model using mySQL to achieve the device security as shown in

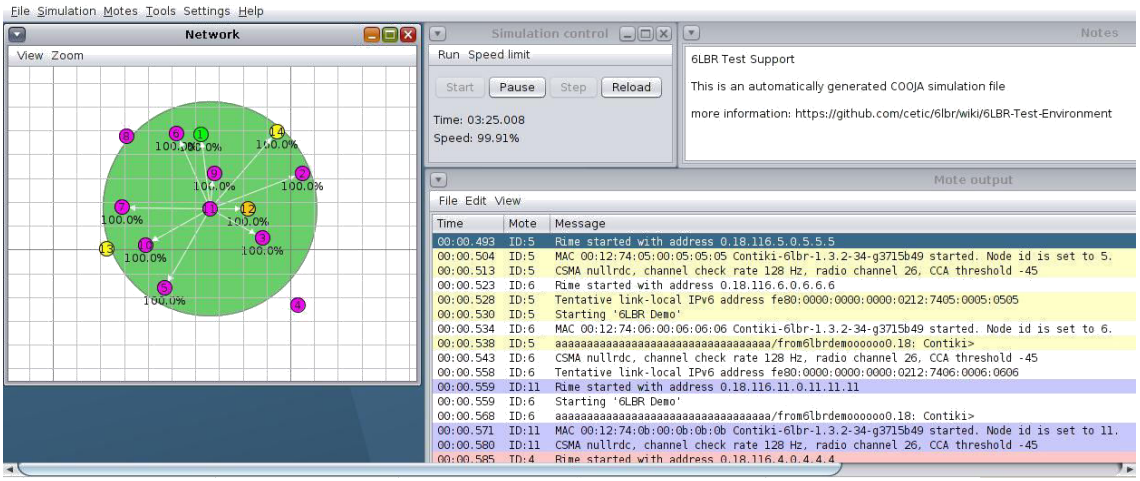


FIGURE 11. Wireless COOJA simulation.

```
Open other console
cd examples/6lbr/test
sudo make launch-6lbr-router-1-cooja
The simulation will start withe the following trace
Fetching MAC address
Fetching MAC address
```

FIGURE 12. Border router for COOJA.

Figure 14. The read and write permissions on the basis of the roles of the users were granted. For the demo purposes, we chose the manual addition of users, roles and permission to the database, but other methods such as deciding roles on the basis of existing social networking websites or existing email ids can be chosen for managing the access control on devices. The system also provides the web interface for the client to access the resources. It is an MVC based server written using technologies like J2EE, HTML, CSS, JSTL.

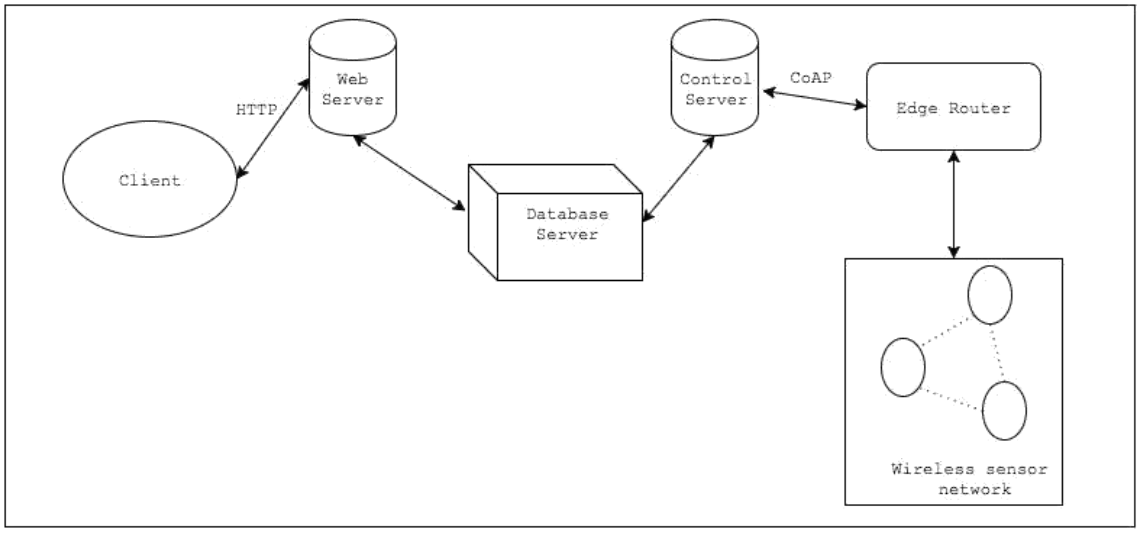


FIGURE 13. System structure.

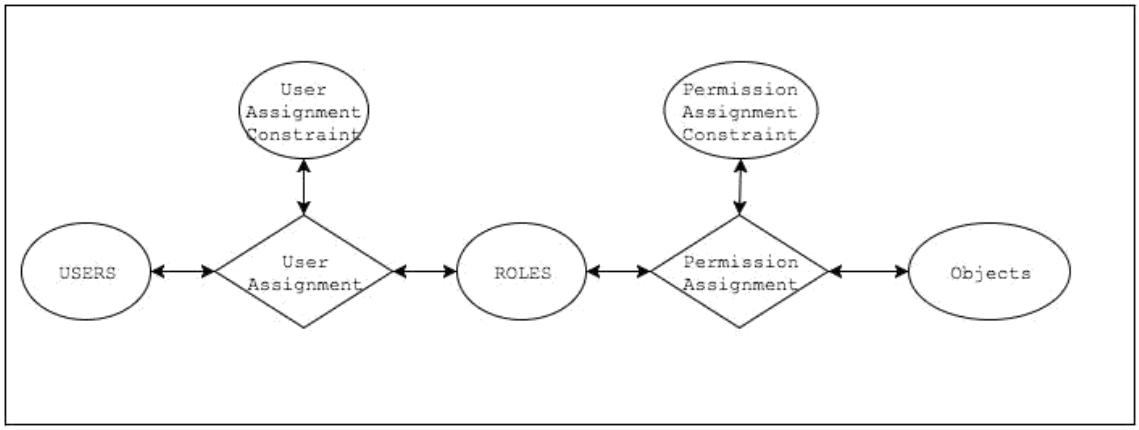


FIGURE 14. RBAC framework.

CHAPTER 5

RESULTS AND DISCUSSION

The implemented system was evaluated in terms of code size, memory usage and power measurement.

Evaluation Techniques

Size and Memory Measurement

We used arm-none-eabi-size, provided by GNU for ARM processor utility, to evaluate the code size and memory usage. The arm-none-eabi-size utility provides us with four data sizes in bytes : text- it shows the read only data and code size in the application, data-it shows the read write data size, bss-it shows the size of data that is zero initialized, dec-it is the sum of above three. The code size can be calculated as the sum of text and data whereas the RAM consumption can be calculated as sum of data and bss.

Packet Overhead

The packet overhead is assessed using the Wireshark-a packet analyzer software. This software can be used to capture packets being transferred between the nodes

Power Measurement

We used powertrace module provided by contiki to estimate the power consumption. We can run the powertrace module on Cooja or with a real device. In order to include powertrace app, it should be added in the Make file of the project. The header file *#include "powertrace.h"* needs to be included in the source file. The command powertrace start(CLOCK SECOND 6) needs to be included in order to

print power profile every six seconds. The powertrace module outputs CPU, LPM, TX and RX. The energy consumption power mW can be calculated as shown in equation (5.1).

$$\frac{Energest\_value\ X\ Current\ X\ Voltage}{RTIMERSECOND\ X\ Runtime} \qquad (5.1)$$

The duty cycle(%) can be calculated as shown in equation (5.2)

$$\frac{Energest\_TX + Energest\_RX}{Energest\_CPU + Energest\_LPM} \qquad (5.2)$$

We also used a multi-meter for the current measurement of the SensorTag after disabling the BLE and net-uart support.

Evalution

Code Size and Memory Usage

We calculated the code size of our application with and without the DTLS as shown in Table 1. The code size of the application without the DTLS library came out to be around 84,365 bytes i.e. around 82 KB and the RAM usage was around 15KB whereas when we calculated it with the DTLS library the code size came out to be

TABLE 1. Memory/Code Size

Cases	Size in KB
Code size without DTLS	82
Code size with DTLS	92
Memory usage without DTLS	15
Memory usage with DTLS	21

Packet Overhead

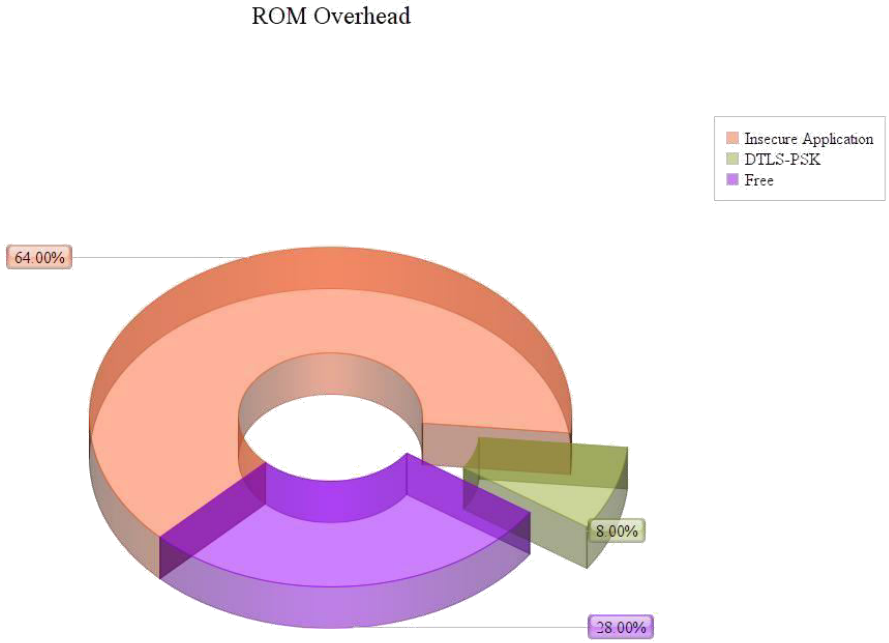


FIGURE 15. ROM overhead

around 92KB and the RAM usage was around 21KB. Figure 15 shows the amount of memory overhead due to DTLS.1 Table 2 shows the fragment length comparison for unsecured CoAP application and secured CoAP application whereas Table 3 depicts the fragment length of the handshake messages. From Table 2, we can conclude that the DTLS using PSK adds upto 29 bytes of overhead to the application.

TABLE 2. Unsecured v/s Secured Fragment Lengths

	Insecure Fragment Length	Secure Fragment Length
Request	25	54
Response	20	52

Power Consumption

The powertrace experiments shown below provide various information. The

TABLE 3. Handshake Fragment Lengths

	Fragment Length(bytes)
Client Hello	82
Hello Verify Request	55
Client Hello	98
Server Hello	38
Client Key Exchange	37
Finished(Client)	50
Finished(Server)	40

	Secured		Unsecured	
	Time(ms)	Energy(microjoules)	Time(ms)	Energy(microjoules)
Request	11.98	1272	3.56	342
Response	12.34	1237	2.13	420

FIGURE 16. CoAP Unsecured v/s Secured energy consumption

energy consumption of secured v/s unsecured CoAP transaction is described in Figure 16. The wide difference can be seen among the values. The DTLS handshake is one of

TABLE 4. DTLS Handshake Flights Energy Consumption

Flights	Energy consumption(microjoules)
1,2	800
3,4	1358
5	26700
6,7	7100

the performance degradation factors. The energy consumption for different flights can be seen in Table 4.

It can be depicted from tables that the power consumption of secured CoAP in a constrained environment is more than double the power consumption in unsecured CoAP environment. The handshake can prove to be a big overhead due to high packet overhead and power consumption.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Conclusion

The purpose of the thesis was to simulate the complete Web of Things architecture. We analyzed some of the existing works and technologies, compared them and chose the suitable ones. We provided an end-to-end approach towards IoT device security by implementing Constrained Application Protocol(CoAP) over Datagram Transport Layer Security(DTLS) in the system. After analyzing the performance of the system in constrained environment, we found that the power consumption in the secured CoAP environment was more than double the power consumption in the unsecured CoAP environment. Also, the DTLS handshake was considered as one of the performance degradation factors because of its high time and energy consumption. A complete web portal for accessing the sensors and their data using the Californium framework was created and Role Based Access Control(RBAC) functionality was implemented.

Recommended Future Work

Although the results presented in the thesis demonstrate the effectiveness of the approach, it could be further enhanced in a number of ways. Some of the areas that need further work are explained below:

DTLS using TLS ECDHE ECDSA WITH AES 128 CCM 8 cipher suite

Because of the memory constraint on the sensortag, we were not able to evaluate the performance of the system using the DTLS in ECC mode. We tried to

implement it on Cooja Simulation Environment, but the implementation failed with two out of seven handshake success rate. Moreover, the time taken for a complete and successful handshake was more than five minutes, which was definitely unacceptable. Therefore, we would like to implement the optimized version of this functionality and then evaluate its performance.

Evaluation of Communication Among Constrained Devices

In this thesis, we have evaluated the communication between a constrained device and unconstrained device. We would like to evaluate the communication among the constrained devices and their performance with DTLS handshakes and data exchanges.

Access Control Implementation at Sensor Level

We have implemented the access control at resource server. However, we would like to implement the role based access control at sensor level. When the client tries to access a resource, DTLS message can include a fixed bit integer number defined for roles with a particular bit set to 0 or 1 based on the access permissions. If the bit is set to 1, the permission can be granted to the particular role.

APPENDICES

APPENDIX A

RESOURCE SERVER

```
#if BOARD_SENSORTAG
  rest_activate_resource(&res_bmp280_temp, "sen/bar/temp");
  rest_activate_resource(&res_bmp280_press, "sen/bar/pres");
  rest_activate_resource(&res_tmp007_amb, "sen/tmp/amb");
  rest_activate_resource(&res_tmp007_obj, "sen/tmp/obj");
  rest_activate_resource(&res_hdc1000_temp, "sen/hdc/t");
  rest_activate_resource(&res_hdc1000_hum, "sen/hdc/h");
  rest_activate_resource(&res_opt3001_light, "sen/opt/light");
  rest_activate_resource(&res_mpu_acc_x, "sen/mpu/acc/x");
  rest_activate_resource(&res_mpu_acc_y, "sen/mpu/acc/y");
  rest_activate_resource(&res_mpu_acc_z, "sen/mpu/acc/z");
  rest_activate_resource(&res_mpu_gyro_x, "sen/mpu/gyro/x");
  rest_activate_resource(&res_mpu_gyro_y, "sen/mpu/gyro/y");
  rest_activate_resource(&res_mpu_gyro_z, "sen/mpu/gyro/z");
  rest_activate_resource(&res_leds, "lt");
  rest_activate_resource(&res_toggle_green, "lt/g");
  rest_activate_resource(&res_toggle_red, "lt/r");
```

FIGURE 17. Adding CoAP Resources to sever.

```
get_hdc_reading()
{
  int value;
  char *buf;
  clock_time_t next = SENSOR_READING_PERIOD +
    (random_rand() % SENSOR_READING_RANDOM);

  if(hdc_temp_reading.publish) {
    value = hdc_1000_sensor.value(HDC_1000_SENSOR_TYPE_TEMP);
    if(value != CC26XX_SENSOR_READING_ERROR) {
      hdc_temp_reading.raw = value;

      compare_and_update(&hdc_temp_reading);

      buf = hdc_temp_reading.converted;
      memset(buf, 0, CC26XX_WEB_DEMO_CONVERTED_LEN);
      snprintf(buf, CC26XX_WEB_DEMO_CONVERTED_LEN, "%d.%02d", value / 100,
               value % 100);
    }
  }
```

FIGURE 18. Fetching temperature value from sensor example.

```
PROCESS_THREAD(cc26xx_web_demo_process, ev, data)
{
  PROCESS_BEGIN();


  init_sensors();

  cc26xx_web_demo_publish_event = process_alloc_event();
  cc26xx_web_demo_config_loaded_event = process_alloc_event();
  cc26xx_web_demo_load_config_defaults = process_alloc_event();

  /* Start all other (enabled) processes first */
  process_start(&httpd_simple_process, NULL);
#if CC26XX_WEB_DEMO_COAP_SERVER
  process_start(&coap_server_process, NULL);
#endif

#if CC26XX_WEB_DEMO_6LBR_CLIENT
  process_start(&cetic_6lbr_client_process, NULL);
#endif
```

FIGURE 19. Starting CoAP server and 6LBR client process.

APPENDIX B

ACCESS CONTROL

```
public boolean setPermission(GroupSensor groupSensor)
{
RetrunUsernameUsingGroupId returnUsename=new RetrunUsernameUsingGroupId();
String usernames[]=returnUsename.returnuserNames(groupSensor.getGroupId());
String permissions[]=groupSensor.getPermission();
String sensorNames[]=groupSensor.getSensorName();
Connection con=DBConnect.returnConnectionObject();
Statement statement = null;
boolean status=true;
for(int j=0;j<sensorNames.length;j++)
{
if(permissions[j].equals("r"))
{
for(int i=0;i<usernames.length;i++)
{
try{
statement = con.createStatement();
status=statement.execute("GRANT SELECT ON sensors."+sensorNames[j]+" TO
"+usernames[i]+";");
return status;
}
catch(Exception e)
{
e.printStackTrace();
}
}
}
else if(permissions[j].equals("rw"))
{
for(int i=0;i<usernames.length;i++)
{
try{
statement = con.createStatement();
status=statement.execute("GRANT ALL PRIVILEGES ON sensors."+sensorNames[j]+"
TO "+usernames[i]+";");
return status;
}
catch(Exception e)
{
e.printStackTrace();
}
}
}
}
return status;
}
```

FIGURE 20. Granting access control permissions to users based on role.

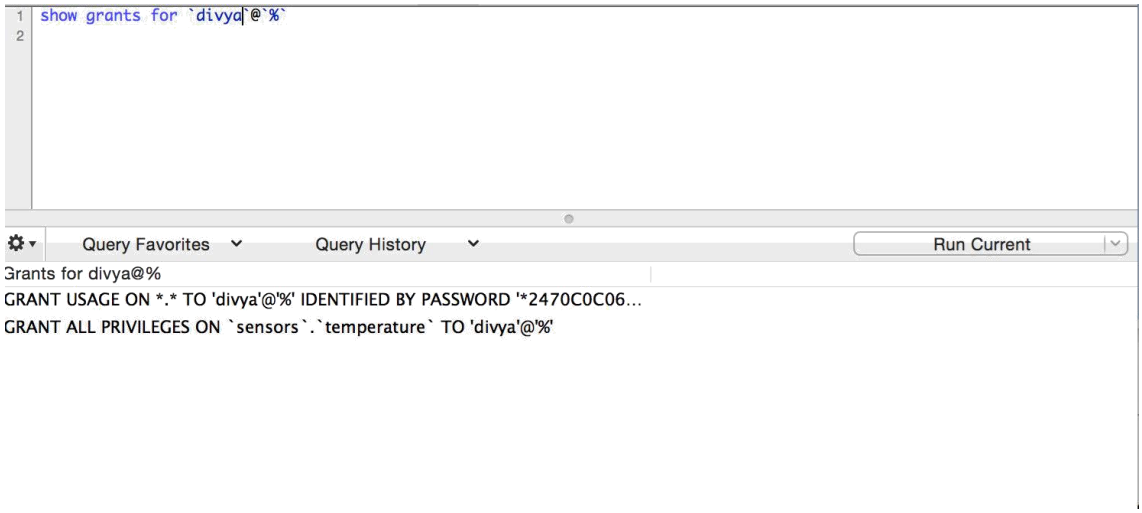
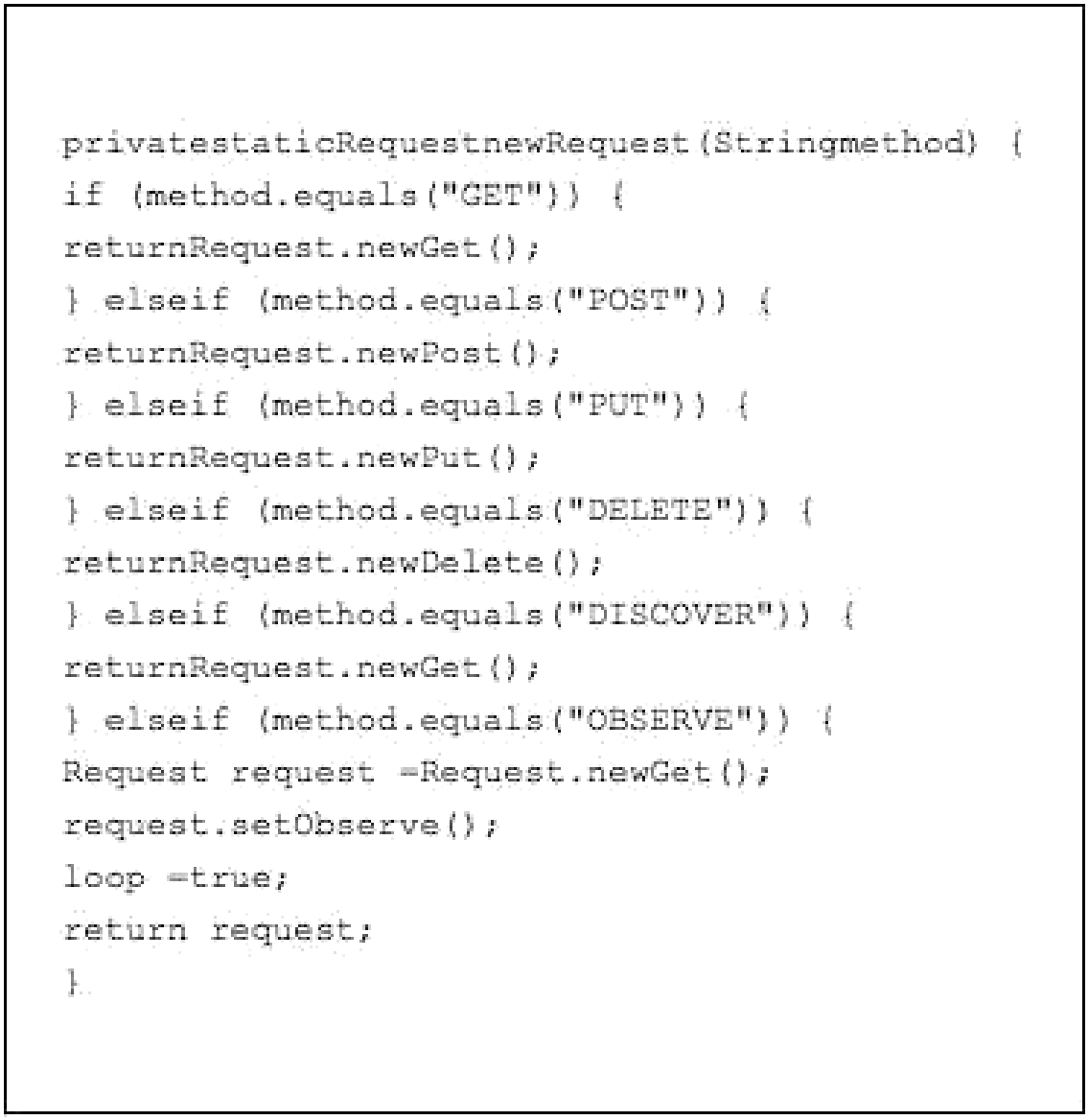


FIGURE 21. Fetch grants to the users.



```
privatestaticRequestnewRequest(Stringmethod) {
if (method.equals("GET")) {
returnRequest.newGet();
} elseif (method.equals("POST")) {
returnRequest.newPost();
} elseif (method.equals("PUT")) {
returnRequest.newPut();
} elseif (method.equals("DELETE")) {
returnRequest.newDelete();
} elseif (method.equals("DISCOVER")) {
returnRequest.newGet();
} elseif (method.equals("OBSERVE")) {
Request request =Request.newGet();
request.setObserve();
loop =true;
return request;
}
```

FIGURE 22. CoAP client methods.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. Singh and A. K. Verma, “Comparative Analysis of IEEE 802.15.4 and IEEE 802.15.6 Standards," paper presented at Sixth International Conference on Computational Intelligence and Communication Networks, 2014.

[2] H. Qarroum, “Awesome IoT," https://github.com/HQarroum/awesome-iot, 2016.

[3] C. Gomez, J. Oller, and J. Paradells, “Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology," Sensors 2012, pp. 11734-11753.

[4] L. Karlsson and J. Assarsson, “Performance of Constrained Wireless Devices in the Internet of Things," student paper, Lund University, Sweden, 2013.

[5] X. Ma and W. Luo, “The Analysis of 6lowpan Technology," Pacific-Asia Workshop on Computational Intelligence and Industrial Application (PACIIA'08), vol. 1, 2008, pp. 963-966.

[6] N. Modadugu and E. Rescorla, “The Design and Implementation of Datagram TLS.," NDSS, 2004.

[7] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, “MQTT-S a Publish/ Subscribe Protocol for Wireless Sensor Networks," IEEE 3rd International Conference of Communication Systems Software and Middleware and Workshops, (COMSWARE 2008), 2008, pp. 791-798.

[8] Texas Instruments, “Ti,” http://www.ti.com/product/CC2650.

[9] Texas Instruments, “Ti," http://www.ti.com/product/CC2531.

[10] Wikipedia, “Contiki,” https://en.wikipedia.org/wiki/Contiki.

[11] ThingsSquare, “Contiki,” http://www.contiki-os.org/.

[12] ARMmbed, “mbed," https://developer.mbed.org/.

[13] github, “cetic," http://cetic.github.io/6lbr/.

[14] M. Kovatsch, M. Lanter, and Z. Shelby, “Californium: Scalable Cloud Services for the Internet of Things with CoAP," International Conference on the Internet of Things (IOT), 2014, vol. 43, pp. 1-6.

[15] M. Kovatsch, Scalable Web Technology for the Internet of Things. PhD thesis, Diss., Eidgenossische Technische Hochschule ETH Zurich, Nr. 22398, 2015.

[16] O. Bergmann, “libcoap," https://libcoap.net/.

[17] A. Nasarre Ramrez, “Internet of Things Implementation with Raspberry Pi," 2014.

[18] M. Welsh, D. Culler, and E. Brewer, “Seda: An Architecture for Well-Conditioned, Scalable Internet Services," ACM SIGOPS Operating Systems Review, vol. 35, no. 5, 2001, pp. 230-243.

[19] Wikipedia, “Cc26xx sw examples," http: //processors.wiki.ti.com /index.php/Cc26xx_sw_examples.

[20] D. Sitenkov, S.-L. Seitz, S. Raza, and G. Selander, “Access Control in the Internet of Things," Master's thesis, 2014.