ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/285236710

Design, Implementation, and Evaluation of 6LoWPAN for Home and Building Automation in the Internet of Things

Conference Paper · November 2015

DOI: 10.1109/AICCSA.2015.7507264

CITATIONS
5

READS
1,922

4 authors:

Son N. Han
Institut National des Télécommunications
22 PUBLICATIONS **547** CITATIONS
SEE PROFILE

Quyet H. Cao
Orange Labs
10 PUBLICATIONS **67** CITATIONS
SEE PROFILE

Bahram Alinia
Institut National des Télécommunications
12 PUBLICATIONS **59** CITATIONS
SEE PROFILE

Noel Crespi
Institut Mines-Télécom
403 PUBLICATIONS **3,053** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Fake News and Bot Detection View project

Collaborative Analytics Platform View project

All content following this page was uploaded by Son N. Han on 01 December 2015.

The user has requested enhancement of the downloaded file.

Design, Implementation, and Evaluation of 6LoWPAN for Home and Building Automation in the Internet of Things

Son N. Han*, Quyet H. Cao*†, Bahram Alinia*, Noel Crespi*
*Institut Mines-Telecom, Telecom SudParis, CNRS UMR 5157, France
Email: {son.han, bahram.alinia, noel.crespi}@it-sudparis.eu
†Orange Labs, France
Email:{quyet.caohuu}@orange.com

***Abstract*—IPv6-enabled low-power wireless personal area networks of smart objects (6LoWPANs) play an important part in the Internet of Things (IoT), especially on account of the Internet integration (IPv6), energy consumption (low-power), and ubiquitous availability (wireless). This paper presents our experience of designing and implementing 6LoWPANs for developing IoT applications, particularly for home and building automation. The performance evaluation provides a comprehensive analysis on several communication aspects between 6LoWPANs and regular IPv6 networks such as energy consumption, network performance, and service communication.**

***Index Terms*—6LoWPAN, Internet of Things.**

I. Introduction

Internet of Things (IoT) involves connecting embedded devices such as sensors, home appliances, health-monitoring devices, and even street lights to the Internet. With about 10 to 15 billion microcontrollers being shipped every year [1], each of which can potentially be connected to the Internet, a huge variety of intelligent and networked devices are becoming available to benefit many application domains (e.g., building automation, healthcare services, smart grids, transportation, industrial automation, and environmental monitoring.) To facilitate this connection, while tackling the energy issue, research and industry have come up over the past decade with a number of advances in low-power microelectronic technology, radio, and networking. These technologies are being engineered by standardization bodies led by Internet Engineering Task Force (IETF) to make them available to the market. The objective is for embedded devices to consume very low energy, become IP-enabled, and to be an integral part of the services on the Internet. These connected devices are referred to as smart objects characterized by sensing, processing, and networking capabilities [2]. IPv6-enabled low-power wireless personal area networks of smart objects (6LoWPANs) are important part in IoT, especially in regard to the ubiquitous availability, energy consumption, and the Internet integration. There are more and more wireless devices available in today's consumer electronics market creating ubiquitous environments around us which are gradually changing our life style. Advantages to the wireless connectivity are many such as the convenience to users, easy deployment, and even for aesthetic aspects. Energy consumption will become especially critical in the time when billions of devices getting connected to the Internet. The energy used for maintaining the connectivity only by current wireless technologies such as Wi-Fi and Bluetooth would use huge amount of energy. Therefore, low-power radio and duty cycling mechanisms are crucially required for manufacturing IoT products. Besides, Internet Protocol (IP) for decades has effectively supported Internet applications such as email, the Web, Internet telephony, and video streaming. Internet Protocol version 6 (IPv6) is expected to accommodate a huge number of entities, predictably enough for billions of objects going to be connected to the Internet in the matter of years. 6LoWPAN is known under several names such as Low-Power Wireless Personal Area Network (LoWPAN) [3], Low-power and Lossy Network (LLN)[4], Constrained Environment [5]. In this paper, 6LoWPAN is used to refer to a network of IPv6, low-power, and wireless smart objects using several IETF standards including Routing Over Low power and Lossy networks (ROLL), Constrained RESTful Environments (CoRE), and DTLS In Constrained Environments (DICE), and IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) which shares the same name but otherwise means a set of standards from IETF 6LoWPAN working group [1]. Figure 1 shows a comparison between typical networking stacks of regular IP networks and 6LoWPAN following 4-layer

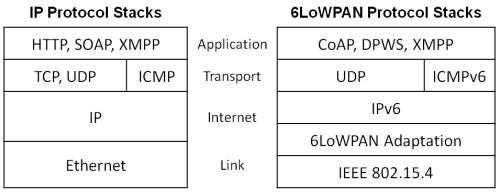


Fig. 1: IP and 6LoWPAN protocol stacks over layers of TCP/IP networking model

[1]In this paper, smart object and 6LoWPAN node are used interchangeably

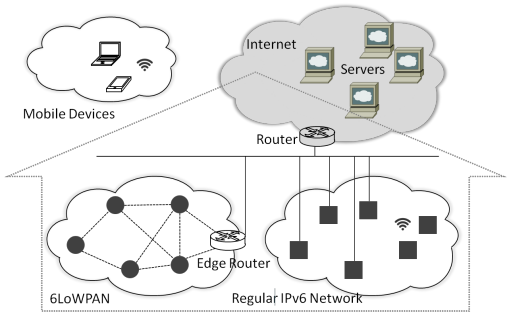


Fig. 2: 6LoWPAN Internetworking Architecture.

TCP/IP model [6]: Link, Internet, Transport, and Applications. The key difference lies at 6LoWPAN adaptation layer, which adds a specific layer and IPv6 header compression before forwarding to regular IPv6 destination. This technology gives the efficient extension of IPv6 into the 6LoWPAN domain, thus enabling end-to-end IP networking features for a wide range of IoT applications. While these technologies are gaining stable status, how they affect the design of many potential intelligent and ubiquitous IoT applications is still rather a new island of discovery. Especially for home and building networks which account for a potentially large portion of IoT application domains. In this paper, we present our experience of designing and developing home and building networks based on networking standards of 6LoWPAN standardized by IETF in CoRE, ROLL, and 6LoWPAN working groups. We implement the design on a set MTM-CM5000-MSP TelosB motes (CM5000) [2] for smart objects and a Raspberry Pi (RPi) [3] for an edge router, some laptop computers for hosts in regular IPv6 network. All are connected to the backbone network of the building. The performance evaluation exhibits how these new networking technologies operate in real-life deployments and where to adapt them to different scenarios.

II. 6LoWPAN Design

A. Internetworking Architecture

The 6LoWPAN internetworking architecture is made up of 6LoWPANs, regular IP networks (IPv4, IPv6), and routers. The overall architecture is presented in Figure 2 in which 6LoWPAN is an IPv6 subnet of smart objects sharing a common IPv6 address prefix (the first 64 bits of an IPv6 address). These smart objects can play the role of host or router to create a mesh network. 6LoWPAN is connected to regular IP networks through an edge router. The edge router forwards data packets between the 6LoWPAN and backbone IPv6, while handling IPv6 compression and neighbor discovery. Communication between 6LoWPAN smart objects and IP hosts in other networks happens in an end-to-end manner, just like between any regular IP nodes. Each 6LoWPAN smart object is identified by a unique IPv6 address, and is capable of sending and receiving IPv6 packets. In Figure 2, the 6LoWPAN smart objects can communicate with either of the regular IPv6 hosts, servers on the Internet, or personal users' devices. Smart objects support ICMPv6 traffic (ping), and use the User Datagram Protocol (UDP) as a transport. Since the payload and processing capabilities of smart objects are extremely limited to save energy, application protocols are designed to use a simple binary format over UDP such as Devices Profile for Web Services (DPWS) [7] and Constrained Application Protocol (CoAP) [5].

B. 6LoWPAN Edge Router

In order to connect 6LoWPAN networks to other IP networks, we use 6LoWPAN edge routers (6EdR). These edge routers are located at the border of the 6LoWPAN performing two essential tasks: adaptation between 6LoWPAN and regular IPv6 networks and routing the IP traffic in and out of the 6LoWPAN. This transformation is transparent, efficient and stateless in both directions. Figure 3 presents our 6EdR architecture consisting of several layers: Network Interfaces (regular IPv6, e.g., Ethernet and low-power, e.g., IEEE 802.15.4), 6LoWPAN Adaptation, Neighbor Discovery, IPv6, IPv6 Routing, Network Management, and Proxy. 6LoWPAN Adaptation Layer is for decompressing frames received from the low-power link [3] using known information about the network and compressing regular IPv6 frames from the regular network interface. This step could be performed in the wireless interface or the edge router driver. Neighbor Discovery is responsible for several configuration tasks such as autoconfiguration of nodes, discovery of other nodes on the link, and maintaining reachability information about the paths to other active neighbor nodes. It includes both IPv6 Neighbor Discovery Protocol (NPD, RFC 4861) and 6LoWPAN Neighbor Discovery (6LoWPAN-ND, RFC 6775). The interface or driver should take care of configuring the stack or adapting relevant neighbor discovery messages between 6LoWPAN-ND and NDP. IPv6 Routing maintains route entries between its interfaces belonging to two different routing domains where most traffic flows are coming from the Internet towards one or more 6LoWPAN nodes, or from LoWPAN nodes towards the Internet. Network Management is one of the core features of any network deployment for managing smart objects on 6LoWPAN. It may use Simple Network Management Protocol (SNMP, RFC 6353). Proxy further adds application layer translation models for transferring request in and out 6LoWPAN.

III. 6LoWPAN Implementation

A. Hardware

We use CM5000 motes equiped with 3 LEDs, a temperature sensor, a humidity sensor, two light sensor, and button sensor as generic smart objects to set up a 6LoWPAN. With several sensors and LEDs, CM5000 can represent many home and building appliances such as a light sensor, a light bulb, a thermostat, a switch, and even a motion sensor. We use a RPi for the edge router with the built-in Ethernet as an IPv6 interface and a CM5000 mote connecting to RPi USB port as a 6LoWPAN interface (IEEE 802.15.4.) Some laptop computers

[2]http://www.advanticsys.com/shop/mtmcm5000msp-p-14.html
[3]http://www.raspberrypi.org/

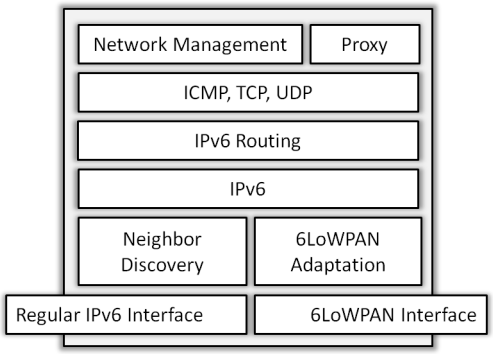


Fig. 3: 6EdR Design

are used to deploy a regular IPv6 networks with Ethernet interfaces connecting to the same router with the edge router.

B. Software

We use the latest update of Contiki OS 3.x (version 2015/02/16) with μIP protocol stack to implement IPv6 networking functionalities for the 6LoWPAN nodes. TI MSP430 toolchain on Ubuntu is used to compile the programs for CM5000 motes. These programs all configure the smart objects to use radio channel 26, ContikiMAC [8] for duty cycling mechanism, router mode to create a mesh network, and MAC addresses to auto generate their IPv6 addresses (e.g., MAC 00:12:74:00:13: cb:2d:a6 for IPv6 aaaa::212:7400:13cb:2da6 address). On top of that, several modules are developed to provide different functionalities to the smart objects such as energy profiling, UDP server, and CoAP server. Figure 4 shows the real hardware configuration of the edge router with two interfaces: IEEE 802.15.4 and Ethernet. Raspbian OS, a Debian-based OS is provided as the platform for the edge router. Its IEEE 802.15.4 interface communicates with the edge router via USB port using Session Initiation Protocol (SIP) protocol. We create a TUN virtual interface to simulate a network device operating on internet layer. This TUN interface works with SIP to apply 6LoWPAN adaptation. We also configure the Raspbian OS as an IPv6 router between two network interfaces Ethernet and TUN. By that, traffic from 6LoWPAN comes to the edge router with IEEE 802.15.4 frames adding compression and 6LoWPAN adaptation in the software, passed to TUN interface and then routed to Ethernet interface to reach regular IPv6 network. For example, when a 6LoWPAN packet is forwarded to the IPv6 network, edge router removes its 6LoWPAN adaptation layer, uncompresses its header, and ensures that global IPv6 source address is used for the outgoing packets. For incoming packets to the 6LoWPAN, edge router adds 6LoWPAN specific adaptation layer and possibly 6LoWPAN IPv6 header compression mechanism and then forwards them to the 6LoWPAN.

IV. Performance Evaluation

We carry out the experiments on the communication between a 6LoWPAN and a regular IPv6 network to observe the quality of the link in several aspects under a real-life

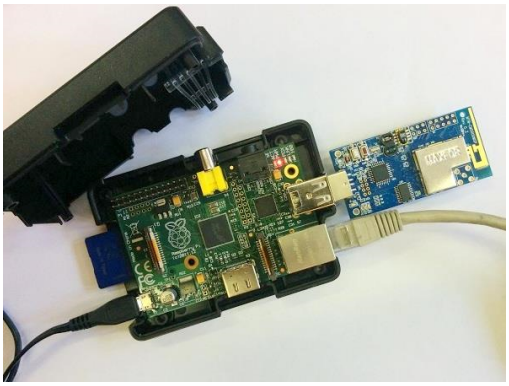


Fig. 4: 6EdR hardware: Raspberry Pi and CM5000 mote as IEEE 802.15.4 interface.

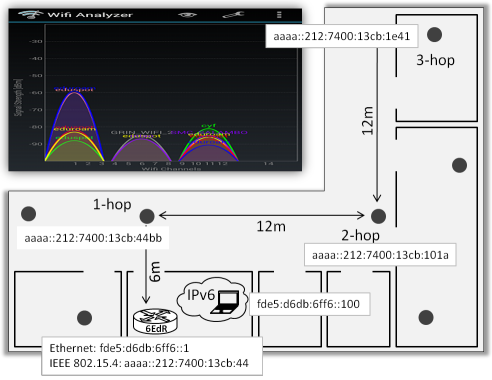


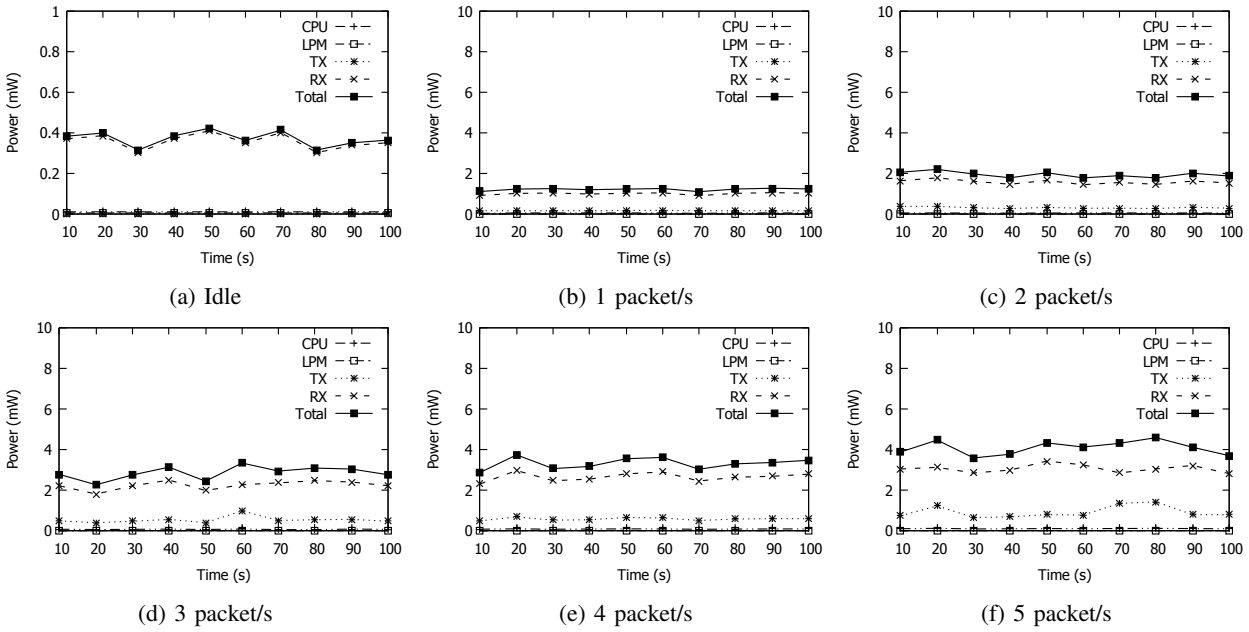
Fig. 5: 6LoWPAN home network setting.

deployment. The deployment takes place in an L-shape office building floor. We deploy the 6LoWPAN with a 6EdR and a number of nodes and a simple IPv6 network with one host. The 6EdR is deployed in one office along with a laptop computer (as a regular IPv6 host), both connected to the same local network via a home and building router. Three nodes are put in 1-hop, 2-hop, 3-hop positions to the edge router as shown in Figure 5. There estimates about 10 Wi-Fi devices operating at the time of the experiment. A screen capture from a Wi-Fi analyzer indicates which channels wireless networks are on and how strong they are. We notice that only *eduroam* and *eduspot* are busy on channel 1, other in mild status which would not affect much on the experiment nodes operating on IEEE 802.15.4 radio channel 26. Network configuration is as follows:

Building Router Linksys E1200, IPv6-enabled
6LoWPAN aaaa::/64
6EdR (Raspberry Pi and CM5000)
- Ethernet: fde5:d6db:6ff6::1 (connected to E1200)
- IEEE 802.15.4: aaaa::212:7400:13cb:44
- Virtual TUN: aaaa::1

Smart objects
- 1-hop node: aaaa::212:7400:13cb:44bb
- 2-hop node: aaaa::212:7400:13cb:101a
- 3-hop node: aaaa::212:7400:13cb:1e41

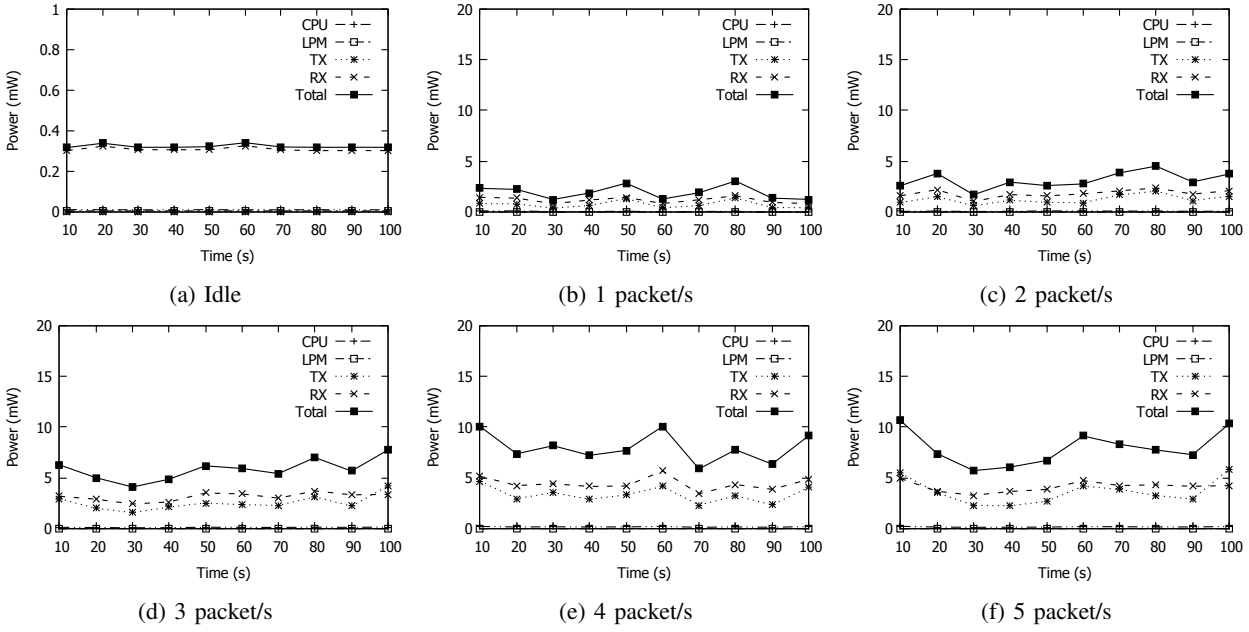
Laptop computer
- Ethernet: fde5:d6db:6ff6::100 (connected to E1200)



(a) Idle (b) 1 packet/s (c) 2 packet/s

(d) 3 packet/s (e) 4 packet/s (f) 5 packet/s

Fig. 6: Energy consumption host mode



(a) Idle (b) 1 packet/s (c) 2 packet/s

(d) 3 packet/s (e) 4 packet/s (f) 5 packet/s

Fig. 7: Energy consumption in router mode

A. Energy Consumption

The first experiment is about energy consumption. We examine how each node in the network consumes energy in two modes (host and router) with five different data rates (1 to 5 packet/s). A smart object is considered to be in *router mode* if it forwards the traffic between other nodes. The *host mode* is when the smart object only communicates with other objects or the edge router without doing any traffic forwarding. We use the power profile Energest [9] in Contiki OS to record the energy consumption in a target object. Energest uses power state tracking to estimate system power consumption and a structure called energy capsules to attribute energy consumption to activities including CPU in active mode (CPU), CPU in standby mode low-power mode (LPM), packet transmissions (TX), and receptions (RX). The power for each activity is calculated by following Formula 1:

$$\frac{Energest\_Value \times Current \times Voltage}{RTIMER\_SECOND \times Runtime} \quad (1)$$

where Energest_value is the value of Energest profile tracked in each activity. Current is the current consumption, which, according to the datasheets of TI CC2420 transceiver and TI MSP430F1611 microcontroller, is 330 μA, 1.1 μA, 18.8 mA, and 17.4 mA for CPU, LPM, TX, and RX respectively. Voltage

is the supply voltage, in this case, 3 V for two AA batteries. RTIMER_SECOND is the number of ticks per second for the RTIMER in Contiki OS, which is 32768. Runtime is the runtime between two Energest track points. The results are shown in Figure 6 for smart objects in *host mode* and in Figure 7 for objects in *router mode*. As can be seen from the graph, the power remains low at 0.4 mW when smart objects are idle, and increases proportionally to the data rate in both *host* and *router mode*.

B. Duty Cycle

The second experiment is to explore radio duty cycle in each 6LoWPAN node. Similar to recording energy consumption, we also use Energest power profile to estimate the duty cycle of each smart object. ContikiMAC radio duty cycling mechanism is enabled in smart objects. It aims to keep their radio transceivers off as much as possible to reach a low power consumption, but wake up often enough to be able to receive communication from their neighbors. Duty cycles are estimated as the percentage of Energest ticks in radio transmission (Energest_TX) and reception (Energest_RX) over the total ticks of the microcontroller in CPU and LPM modes (Energest_CPU, Energest_LPM) over a period of time (10 seconds) by following Formular 2:

$$\frac{Energest\_TX + Energest\_RX}{Energest\_CPU + Energest\_LPM} \tag{2}$$

Figure 8 and Figure 9 depict duty cycles of a smart object in *host mode* and *router mode* with 5 different data rates of 1, 2, 3, 4, and 5 packet/s. In general, duty cycle of a smart object in *host mode* is lower and more stable than in *router mode*. Forwarding data packets apparently requires radio to be more waken-up then only receiving data. When smart objects are idle (or in sleep mode, but still wake up frequently enough to maintain the connectivity), the duty cycle remains fairly low about 0.3 percent in the *host mode* and 0.6 percent in the *router mode*. Duty cycle increases constantly over the change of data rate from 1 to 5 packet/s.

C. Network Performance

In the third experiment, we send 100 Internet Control Message Protocol version 6 (ICMPv6, RFC4443) packets from a regular IPv6 host to different smart objects in the 6LoWPAN and wait for the echo responses to record some network parameters such as packet loss, round-trip time, and time-to-live to calculate packet delivery ratio (PDR), end-to-end delay, and data transfer rate. The experiment is to send three sets of packets to three types of 6LoWPAN nodes: 1-hop, 2-hop, and 3-hop. Each set is carried out in 5 different data rates from 1 to 5 packet/s.

1) Radio Signal Strength: We first carry out a supplementary experiment between two CM5000 motes to measure Received Signal Strength (RSSI) and Link Quality Indication (LQI) between two nodes to access the IEEE 802.15.4 signal strength in CC2420 transceivers. The CM5000 devices are programmed to transmit and receive 802.15.4 wireless beacons. We maintain the active connection (IPv6) between two nodes and record RSSI and LQI over distances ranging from 3 and 21 m with steps of 3 m. Figure 10a shows the experiment results indicating RSSI in a good condition within the range of 21 m and LQI remains stable at 108.

2) Packet Delivery Ratio: As illustrated in Figure 10b, PDR gets very high rate of 98 percent for 1-hop nodes and slightly drops to 86 percent when data rate reaches the highest rate among the tests of 5 packet/s. For 2-hop and 3-hop nodes, PDR is lower through out the experiment fluctuating from 45 to 80 percent.

3) End-to-End Delay and Data Transfer Rate: Figure 10c shows the end-to-end delay slightly increases when more packets come back and forth between nodes, it however remains very low and not much diverse between different types of nodes (1-hop, 2-hop, and 3-hop), ranging from 30 to 60 ms. These figures are considered transparent to the communication. Accordingly, data transfer rate shows a similar pattern with 25 kbit/s, 15 kbit/s, and 10 kbit/s for 1-hop, 2-hop, and 3-hop nodes respectively.

D. Service Communication

When it comes to the development of IoT applications, apart from the IP networking infrastructure, provisioning smart object services is an essential issue. Developers expect APIs that they can integrate new features and create new functionalities to their application. While, it is most likely that future intelligent and ubiquitous applications reside on the Web and interact with a plethora of existing Web APIs. Notably, CoAP is designed exclusively for smart objects to replace HTTP and can be easily translated to HTTP for a transparent integration with the Web, while meeting the smart object requirements such as multicast support, very low overhead, and publish/subscribe model. Since CoAP is not native to the Web protocols, a CoAP/HTTP proxy is a common approach to provide HTTP-based APIs for CoAP services.

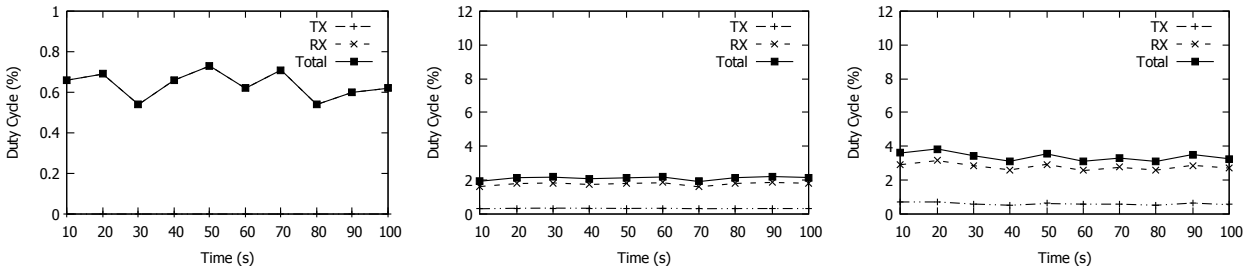
We therefore carry out the fourth experiment on service communication between a Web application and smart objects using CoAP and HTTP protocols (via CoAP/HTTP proxy). We mainly use CoAP Californium library [10] to implement the Web application. Figure 11 presents the request/response message sizes and latency of CoAP and HTTP transactions. CoAP messages apparently smaller than HTTP due to the the use of simplified headers compare to HTTP headers though the difference is at hundred kb. The round-trip time of CoAP and HTTP request are not much different and considered to be transparent to user's experience.

V. Discussion and Lessons Learned

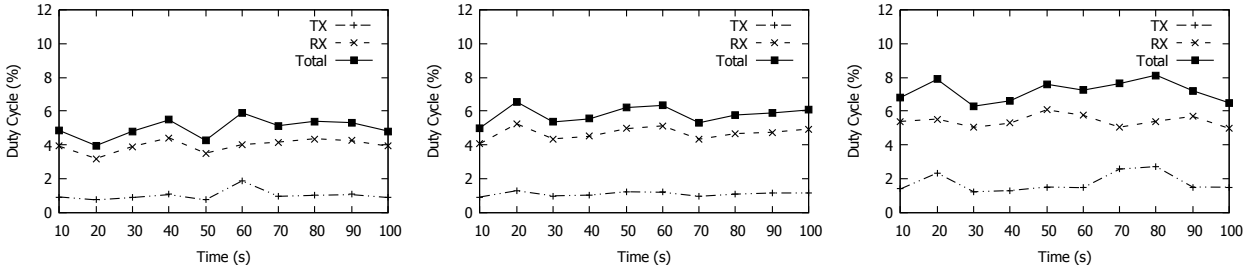
A. Energy Consumption

Based on the average capacity of an AA battery is 2500 mAh and nominal voltage is 1.5 V, we can estimate the battery life for smart objects to maintain the connectivity (using duty cycling) as following Formula 3

$$\frac{2500mAh \times 1.5V \times 2}{0.37mW \times 24h \times 365days} = 2.304years \tag{3}$$

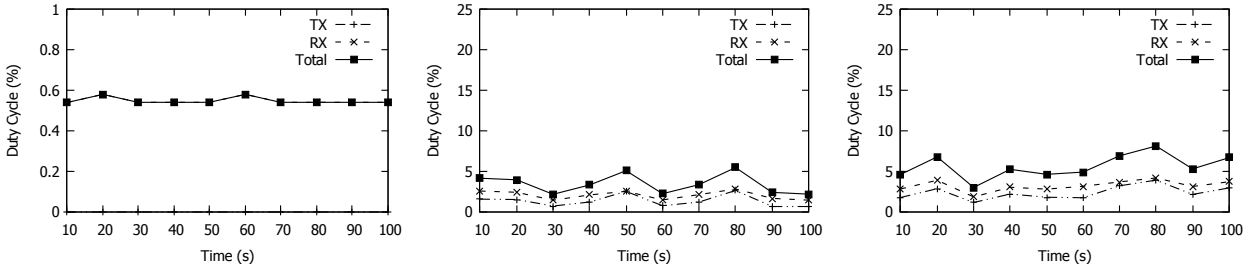


(a) Idle (b) 1 packet/s (c) 2 packet/s

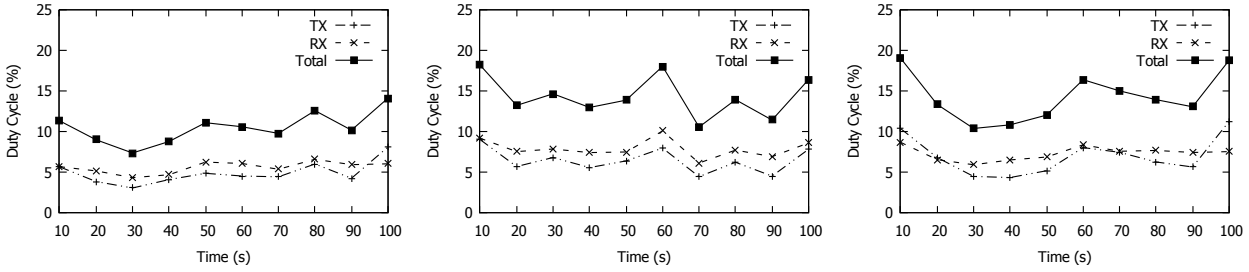


(d) 3 packet/s (e) 4 packet/s (f) 5 packet/s

Fig. 8: Duty cycle in host mode

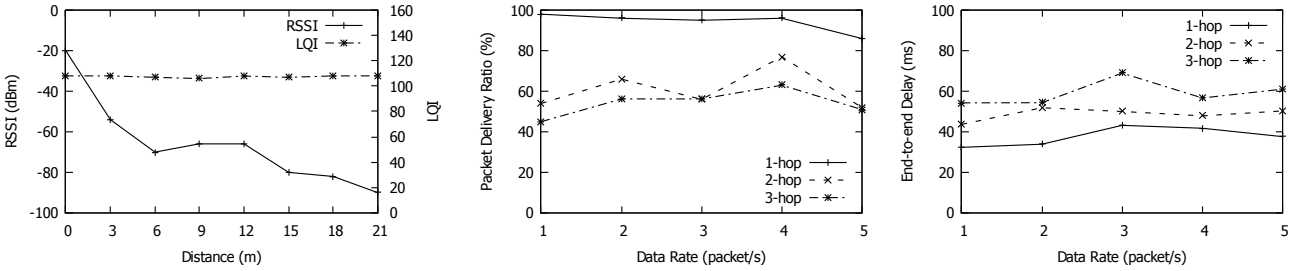


(a) Idle (b) 1 packet/s (c) 2 packet/s



(d) 3 packet/s (e) 4 packet/s (f) 5 packet/s

Fig. 9: Duty cycle in router mode



(a) Radio RSSI and LQI (b) Packet Delivery Ratio (c) End-to-end Delay

Fig. 10: Network Performance

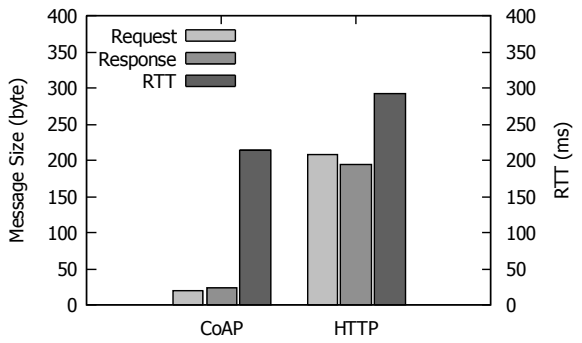


Fig. 11: CoAP vs HTTP message size and latency

where 0.37mW is the average power of smart objects in *host mode* when idle.

The duration of 2.3 years can be considered to be the very long for just only two AA batteries to maintain the connectivity. Similarly, Table I illustrates the estimated battery life of smart objects in some cases: idle (duty cycling) and continuously sending packets with the data rate from 1 to 5 packet/s in two modes (*host* and *router*). With the average duty cycle remains lower than 0.5 percent when objects are idle, the connectivity of 6LoWPANs is still maintained meanwhile energy consumption is kept minimal. Even in case of continuously sending data with very high data rate of 5 packet/s, two AA batteries can provide enough power for about 2.5 months.

TABLE I: Battery powered smart object lifetime.

Data rate	**Lifetime (year)**	
(packet/s)	**Host**	**Router**
Idle	2.303757013	2.64475527
1	0.7060081564	0.442350067
2	0.4414037119	0.271677592
3	0.3000770222	0.146784071
4	0.2581720706	0.107174921
5	0.209452065	0.107788434

B. Contiki OS 3.x and Network Performance

In 1-hop communication, the PDR show very high value of 98 percent, almost at theoretical PDR of IEEE 802.15.4 radio. With more than 1-hop communication, there's an obvious trend of much lower PDR. This is identified as the result of the RPL routing protocol. More investigation is expected to figure out the cause of the packet loss. The data transfer rate at around 25 kbit/s is considered low-rate due to the sacrifice of hardware for the sake of energy consumption. Many IoT devices such as home appliances and sensor nodes only transfer control data with few bytes then this rate is adequate for most IoT applications containing relatively simple service communication. When considering new application ideas, system designers are expected to take into account the transfer rate to make a right choice for the network deployment.

The operation of uIP networking stacks in Contiki OS 3.x appears reliable in our intensive experiment with packets sending for a period of 24 hours. Compared to previous releases, our experience with Contiki OS 3.x indicates that IP performance has been improved considerably. Besides, there are several useful libraries with Contiki OS such as Erbium CoAP, Web server, file system, and Shell. Contiki OS programming experience is very effective with protothreads for multi-threading and event-driven applications. Our experience suggests Contiki OS is very robust and can be the universal operating system for smart objects.

C. Current IPv4 Infrastructure

Even though IPv6 is an ideal addressing space for future Internet but the shift to IPv6 is still happening at a slow pace accounting for only 5 percent of the worldwide Internet traffic, according to Cisco 6lab [4]. Smart objects have just arrived but already bear a full support of IPv6 rather than IPv4 (there is apparently no IPv4 adaptation layer for IEEE 802.15.4 alike 6LoWPAN). A backward integration appears to be a temporary problem during the transition time from IPv4 to IPv6. Some basic transition mechanisms between IPv4 and IPv6 systems have been proposed and applied throughout the Internet (RFC 4213). However, the use of such techniques for smart objects and 6LoWPAN may costly and double the effort to use IP technologies for smart objects. Furthermore, in contrast to conventional computer networks on the Internet providing several services such as e-mail, telephony, and Web, smart object services tend to use in ubiquitous applications that require application level interface rather than raw IP services. Therefore, proxy can be a fair solution on current Internet infrastructure that doesn't change the backbone of the network and provides a seamless interface for IoT applications.

D. Web Services

There are several candidate protocol for application layer in IoT including HTTP, CoAP, DPWS, XMPP, MQTT, and AMQP. Among which, DPWS and CoAP are mostly close to common Web architecture aiming to bring functionalities of smart objects (data and events) to the Web in the form of services. By following Web design principles (REST, SOA), these services can acquire open Web standards to enable them to understand the Web languages and protocols, denoted as smart object services.

CoAP follows REST architectural style, compromising a minimal subset of REST along with mechanisms of resource discovery, subscription/notification, and security measures for smart objects. It is similar to HTTP and can be easily translated to HTTP for a transparent integration with the Web, while having very low overhead. It also supports multicast and publish/subscribe model. The CoAP protocol provides a technique for discovering and advertising resource descriptions via CoAP endpoints using CoRE Link Format [11] of discoverable resources. As standardized by IETF, CoAP is

[4]http://6lab.cisco.com/

showing suitable for smart objects as well as getting attention from the community. There are many CoAP implementations available not only for smart objects (e.g., Erbium [5] for Contiki OS, *libcoap* for TinyOS, and SMCP [6] for embedded systems) but also for powerful servers (e.g., Java Californium [7]), Web browser (e.g., Copper [8]), and mobile platform (e.g., nCoAP). The Erbium implementation, according to our experiments, exposes very low overhead and supports well multicast as well as publish/subscribe model. This protocol is showing an excellent choice to meet event-driven requirements from IoT application. A secure mechanism for CoAP transaction is expected to explored more to make it widely usable in real-life applications.

DPWS, on the other hand, is the lightweight version of W3C Web Service [12] in addition to new features such dynamic discovery and event notification. Even though DPWS use XML-based SOAP envelopes (something considered bulky), our experiment shows that it can be implemented on top of IP protocol stack to (even) highly resource-constrained smart object such as sensor nodes (thanks to uDPWS [9] and Contiki OS). The request and response messages are relatively large compared to HTTP or CoAP but still well operate on very limited node. Besides, in smart objects with higher computing power and memory such as home appliances and office equipments, DPWS can perform in its best to enable secure translations between smart objects and applications.

E. Deployment

From the experiment results, we look into some deployment issues such as how large 6LoWPAN coverage can be in typical premises and how difficult the deployment can be when it comes to mass production.

1) IEEE 802.15.4 Radio Range: Since the radio signal is considerably strong at 15 m in reality, the range of the network is considered sufficient to several homes and buildings. 1-hop 6LoWPANs which only consist of smart objects in the radio range of the edge router can cover the area of 707 m2 in good radio signal. That area can comfortably cover typical 2-storey houses. Table II shows more details about the estimated ranges in different facilities that 6LoWPAN can give healthy radio coverages. In most cases of average houses and offices, IEEE 802.15.4 can comfortably maintain a stable connectivity.

TABLE II: IEEE 802.15.4 Radio Range.

Node Type	Range (m)	Area (m2)	Typical facilities
1-hop	15	707	large, two-storey houses
2-hop	30	2827	medium building floors
3-hop	45	6362	large building floors

[5]http://people.inf.ethz.ch/mkovatsc/erbium.php
[6]https://github.com/darconeous/smcp/
[7]http://people.inf.ethz.ch/mkovatsc/californium.php
[8]http://people.inf.ethz.ch/mkovatsc/copper.php
[9]http://ws4d.org/udpws/

2) Installation: IoT application thus far is frequently considered high cost and difficulty to deployment. However, with the presented design, the deployment of 6LoWPAN such as for home and building networks appears to be easy and intuitive, in the same way to conventional IP networks. The edge router hardware and software can be developed very fast and at low-cost using current advances in micro-electronics and radios (equivalent to a single computer board plus a 802.15.4 radio module). Besides, services of smart objects can seamlessly reside on the Web by implementing application protocols such as CoAP, DPWS, and protocol proxy for HTTP. By that, the development model from developers' point of view virtually remains the same, which will stimulate more the adoption of IoT applications. Furthermore, the installation of smart objects in 6LoWPAN is zero-configuration, which doesn't require any additional commissioning device (e.g., a laptop computer). In other words, a smart object can obtain an address and join the 6LoWPAN on its own, without human intervention.

VI. Conclusion

We have presented our design, implementation, and evaluation of a 6LoWPAN for home and building networks. The practical lessons learned from real-life implementations, especially in terms of networking metrics, provide a new experience in the practicality of the IoT protocol stack. These lessons also act as a reference for the adoption of the IoT networking technology in other IoT systems. Security features are work in progress with the ITEF new DTLS in Constrained Environments (DICE) working group is currently considered to include Datagram Transport Layer Security [13] for smart objects.

References

[1] M. Roberti, "The internet of things revisited," *RFID Journal*, May 2010.
[2] J.-P. Vasseur and A. Dunkels, *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann, 2010.
[3] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of ipv6 packets over ieee 802.15. 4 networks," IETF, Tech. Rep. RFC 4944, 2007.
[4] T. Winter, P. Thubert, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. Vasseur, "Rpl: Ipv6 routing protocol for low power and lossy networks," IETF, Tech. Rep. RFC 6550, 2012.
[5] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (coap)," IETF, Tech. Rep. RFC 7252, 2014.
[6] R. Braden, "Requirements for internet hosts-communication layers," IETF, Tech. Rep. RFC 1122, 1989.
[7] "Devices Profile for Web Services Version 1.1," OASIS, Tech. Rep., Jul. 2009.
[8] A. Dunkels, "The contikimac radio duty cycling protocol," *Swedish Institute of Computer Science report*, 2011.
[9] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *Proceedings of the 4th workshop on Embedded networked sensors*. ACM, 2007, pp. 28–32.
[10] M. Kovatsch, M. Lanter, and Z. Shelby, "Californium: Scalable cloud services for the internet of things with coap," in *Proceedings of the 4th International Conference on the Internet of Things (IoT 2014)*, 2014.
[11] Z. Shelby, "Constrained restful environments (core) link format," IETF, Tech. Rep. RFC 6690, 2012.
[12] "Web Services Architecture," W3C, W3C Working Group Note, Feb. 2004.
[13] E. Rescorla and N. Modadugu, "Datagram transport layer security version 1.2," IETF, Tech. Rep. RFC 6347, 2012.

View publication stats