Far East Journal of Electronics and Communications

http://www.pphmj.com
http://dx.doi.org/10.17654/EC017040915
Volume 17, Number 4, 2017, Pages 915-921 **ISSN: 0973-7006**

IoT THEORETICAL TO PRACTICAL: CONTIKI-OS AND ZOLERTIA REMOTE

**Yousaf Bin Zikria, Rashid Ali, Rojeena Bajracharya, Heejung Yu*
and Sung Won Kim**

Department of Information and Communication Engineering
Yeungnam University
South Korea
e-mail: yousafbinzikria@ynu.ac.kr
rashin@ynu.ac.kr
rojeena@ynu.ac.kr
heejung@yu.ac.kr
swon@yu.ac.kr

Abstract

Things constitute of all the networked devices that can communicate with each other. Internet of things (IoT) is the future. The realism of IoT depends on the standardization, supported operating systems (OS) and devices. Contiki is an open source operating system for the IoT that follows the Internet standards and supports many hardware platforms. The objective of this paper is to build IoT test bed to test IPv6 motes traffic over low power wireless personal area network to the IPv4 server using the MQ telemetry transport protocol. IoT realization is done using Contiki-OS, Zolertia remote sensors, virtual

Received: March 13, 2017; Accepted: April 30, 2017

Keywords and phrases: IoT, Contiki, Zolertia remote, border router, NAT64, MQTT, MongoDB.

A preliminary version of this paper was presented at the ICIDB-2016, Seoul, South Korea.

*Corresponding author

Communicated by Gyanendra Prasad Joshi

Linux machine acting as a border router and it provides NAT64 conversion as well. Further, the remote server is configured as a mosquito MQTT server with MongoDB database.

I. Introduction

One of the key challenges of Internet of things (IoT) lies in light weight constrained environments. IoT term is first used by Kevin Ashton in 1999 [1]. Contiki-OS [2] guarantees a rich enough execution environment to fulfill the requirements of strict constrained devices. Table I lists key features of the Contiki-OS.

Table I. Contiki-OS features [3]

Features	Contiki-OS
Memory allocation	Few kilobytes
Full IP networking	UDP [4], TCP [5], HTTP [6], 6lowpan [7], RPL [8], CoAP [9]
Dynamic module loading	Loading and linking at run time
Simulator	Cooja
Hardware platforms	8051, MSP430, AVR
Coffee flash file system	Devices with external flash memory chip

In the literature, the complete guideline to build the IoT test bed to communicate between IPv6 and IPv4 is not provided. Therefore, this paper focuses on building and testing the network architecture that constitutes of IPv6 over low power wireless personal area networks (6LoWPAN) with Zolertia remote sensors [10]. Further, an IPv4 remote mosquito MQ telemetry transport (MQTT) [11] server with open source MongoDB NoSQL database platform is used to store the MQTT messages. Moreover, Linux machine configures as a border router and NAT64 uses 6lbr [12] to act as a bridge between IPv6 traffic and IPv4 server.

This paper is organized as follows: Section II discusses in detail the network architecture, installation, configuration and testing. Finally, Section III concludes the paper.

II. Network Architecture

The network architecture is shown in Figure 1. It consists of 3 main parts; 6LoWPAN, border router and NAT64, and a MQTT server with the database. IPv6 packets are destined for the IPv4 server. Hence, the network address translation (NAT) is required along with the border router. A 6LoWPAN border router connects the 6LoWPAN devices to the Internet. Moreover, it is responsible for handling traffic to and from the IPv6 and 802.15.4 [13] interfaces. NAT64 is an IPv6 transition mechanism to facilitate the communication between IPv6 and IPv4 hosts using NAT. We use the official Contiki-OS virtual machine [14] to kick start our deployment.

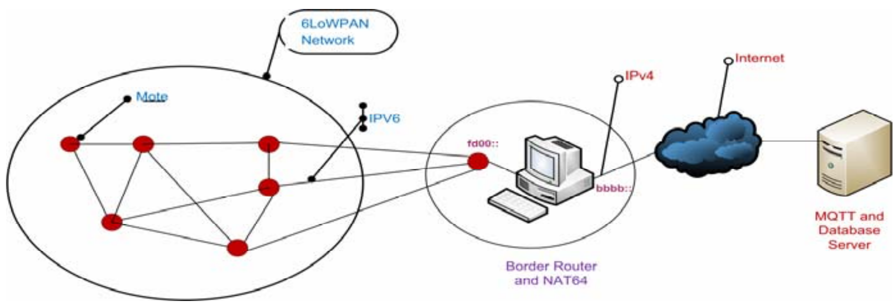

Figure 1. Network architecture.

(a) 6LoWPAN network

Zolertia remote sensors are the state of art IoT devices. Table II lists the features of remote. Figure 2 represents the remote sensors. We need at least two remote sensors to establish the 6LoWPAN network. One remote sensor is programmed as a slip radio. While the other programs as a MQTT client. The remote sensor flashing steps are mentioned in Figure 3. We need to specify the NAT64 address of the MQTT server in the project file of MQTT client for successful connectivity.

Table II. Zolertia remote features [10]

Features	**Zolertia remote**
Hardware	CC2538 ARM Cortex-M3
ROM	512 KB
RAM	32 KB
Radio interface	2.4Ghz IEEE 802.15.4, CC1200 868/915Mhz
Form factor	2 times smaller than an Arduino
Power consumption	Ultra low-power, 300% less than WiFi devices
Battery charger	Built-in battery charger
External storage	Micro-SD card
Compatibility	Raspberry Pi, Beagle Bone, Intel Edison, Any Arduino sensor and actuators
Supported OS	Contiki, RIOT, OpenWSN

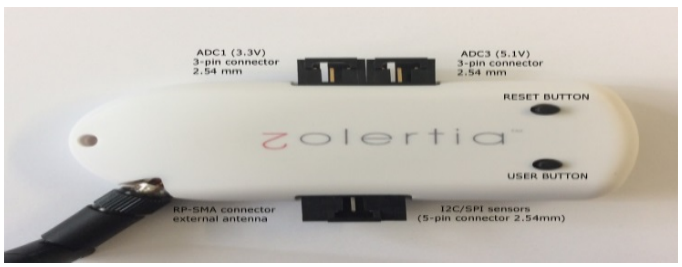

Figure 2. Zolertia remote.

```
cd home/user/contiki/examples/ipv6/slip-radio/
make TARGET=zoul savetarget
BOARD=remote make slip-radio.upload PORT=/dev/ttyUSB0
BOARD=remote make login PORT=/dev/ttyUSB0
cd /user/contiki/examples/mqtt
BOARD=remote make mqtt-example.upload PORT=/dev/ttyUSB1
BOARD=remote make login PORT=/dev/ttyUSB1
```

Figure 3. Flashing Zolertia remote sensors.

(b) Border router and NAT64

We use the guest virtual machine with NAT enabled on the Internet enabled host machine with the public IPv4 address. 6lbr is a deployment ready 6LoWPAN border router and NAT64 solution based on Contiki-OS.

Figure 4 illustrates detail installation and configuration steps. Afterwards we will see bridge, ethernet, tap interfaces and 6lbr web interface.

```
cd /home/user/Downloads
apt-get install libncurses5-dev bridge-utils
git clone https://github.com/cetic/6lbr
cd 6lbr
git submodule update --init --recursive
cd examples/6lbr
make all #all_native for version < 1.4
make plugins
make tools
sudo make install
sudo make plugins-install
sudo update-rc.d 6lbr defaults
cd /etc/6lbr/
sudo gedit 6lbr.conf
MODE=ROUTER
RAW_ETH=0
BRIDGE=1
DEV_BRIDGE=br0
DEV_TAP=tap0
DEV_ETH=ens33
RAW_ETH_FCS=0
DEV_RADIO=/dev/ttyUSB0
BAUDRATE=115200
LOG_LEVEL=3
sudo service 6lbr restart
sudo service 6lbr status
sudo gedit /etc/network/interfaces
auto lo
iface lo inet loopback
auto br0
iface br0 inet dhcp
bridge_ports ens33
sudo /etc/init.d/networking restart
ifconfig
http://[bbbb::100]
Edit Configuration Global Settings
Channel: 26
IP Configuration Prefix: fd00::
IP Configuration 6LoPWAN context 0: fd00::
IP Configuration Address autoconfiguration: on
Eth Network prefix: bbbb::
Eth Network Address autoconfiguration: off
IP64 IP64: on
IP64 DHCP: on
RA Daemon RA Daemon: active
Click Submit and it will reboot the 6lbr daemon
ping google.com, open http://[bbbb::100]
Connect slip Mote to Linux host
Turn on Mqtt Client mote
Click Sensor Tab to see the IPv6 Motes
```

Figure 4. Border router and NAT64 configuration.

(c) MQTT server and MongoDB

We use the mosquito MQTT server and MongoDB [15]. The unabridged installation and configuration steps are shown in Figure 5.

```
wgethttp://mosquitto.org/files/source/mosqu
itto-1.4.2.tar.gz
cd mosquitto-1.4.2/
sudo gedit config.mk
WITH_WEBSOCKETS:=yes
make
make install
sudo make install
sudo cp mosquitto.conf /etc/mosquitto/
sudo gedit /etc/mosquitto/mosquitto.conf
port 1883
protocol mqtt
listener 9001
protocol websockets
sudo apt-key adv --keyserver
hkp://keyserver.ubuntu.com:80 --recv
EA312927
echo "deb
http://repo.mongodb.org/apt/ubuntu
"$(lsb_release -sc)"/mongodb-org/3.2
multiverse" | sudo tee
/etc/apt/sources.list.d/mongodb-org-3.2.list
sudo apt-get update
sudo apt-get install -y mongodb-org
```

Figure 5. MQTT and MongoDB configuration.

(d) Testing

Connect the slip radio mote to the border router. Turn on the remote MQTT client. Reset the MQTT client mote and it connects with the remote MQTT server through border router and the NAT64 Linux machine. Hereafter, it publishes the sensor data to the MQTT server. Figure 6 depicts a successful connection and publishing data to MQTT server.

Figure 6. Publishing on MQTT server.

III. Conclusion

IoT real time deployment is a challenging task. The literature lacks the practical IoT deployment scenario for research and experimentation. Therefore, this paper provides detail network architecture to deploy the IoT scenario. Moreover, we listed in detail all the network components, installation and configuration steps.

Acknowledgment

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning, Korea, under the ITRC (Information Technology Research Center) support program (IITP-2016-R2718-16-0035) supervised by the IITP (National IT Industry Promotion Agency).

References

[1] M. Weiser, The computer for the 21st century, Sci. Amer. 265 (1991), 66-75.

[2] A. Dunkels, B. Grönvall and T. Voigt, Contiki - a lightweight and flexible operating system for tiny networked sensors, Proceedings of the First IEEE Workshop on Embedded Networked Sensors, Tampa, Florida, USA, November 2004.

[3] http://www.contiki-os.org/ (accessed on October 28, 2016).

[4] J. Postel, RFC 768: user datagram protocol, August 1980.

[5] J. Postel, RFC 793: transmission control protocol, September 1981.

[6] T. Berners-Lee, R. Fielding and H. Frystyk, RFC 1945: hypertext transfer protocol -- HTTP/1.0, May 1996.

[7] G. Montenegro, N. Kushalnagar, J. Hui and D. Culler, RFC 4944: transmission of IPv6 packets over IEEE 802.15.4 networks, September 2007.

[8] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur and R. Alexander, RPL: IPv6 routing protocol for low-power and lossy networks, RFC 6550, Internet Engineering Task Force RFC 6550, March 2012.

[9] Z. Shelby, K. Hartke and C. Bormann, RFC 7252: the constrained application protocol (CoAP), June 2014.

[10] http://zolertia.io/product/hardware/re-mote (accessed on October 28, 2016).

[11] https://mosquitto.org/ (accessed on October 28, 2016).

[12] http://cetic.github.io/6lbr/ (accessed on October 28, 2016).

[13] M. McInnis, Editor-in-Chief, 802.15.4 – IEEE Standard for Information Technology, Institute of Electrical and Electronic Engineers, New York, 2003.

[14] http://www.contiki-os.org/start.html (accessed on October 28, 2016).

[15] https://www.mongodb.com/ (accessed on October 28, 2016).