ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)

https://doi.org/10.17559/TV-20180528134139
Original scientific paper

Container-Based Virtualization for Bluetooth Low Energy Sensor Devices in Internet of Things Applications

Deniz TAŞKIN, Cem TAŞKIN, Selçuk YAZAR*

Abstract: Internet of Things (IoT) has become a continuously growing concept with the developments of ubiquitous computing, wireless sensor networks (WSN). With the industry 4.0 revolution, all production activities such as logistics, finance, agriculture, energy and almost all the service and infrastructure applications used by people in the cities we live in will undergo a major change within the IoT paradigm. In this study, a prototype model has been developed and its performance is investigated. Our prototype model can reach the advertisement data of Bluetooth Low Energy sensor devices by using container-based virtualization technology and directly working at layer 2 (L2) of Transmission Control Protocol/Internet Protocol (TCP/IP). Virtualization mechanism for the sensor devices could help to exchange context-aware information with Internet Protocol Version 6 (IPv6) structure. Also with virtualization may emerge interoperable sensor node platforms of heterogeneous environments from different vendors.

Keywords: application virtualization; bluetooth; Internet of Things (IoT); ip networks; wireless sensor networks

1 INTRODUCTION

The core concept of IoT is to enable ubiquitous computing with unique addressing of objects, to store and exchange information without or less human interaction. Assuming that these "things" especially those equipped with sensors and actuators used in various production areas, will have direct Internet access and share their information with IPv6, the need for access and management of these approximately 50 billion of devices [25] will become a huge challenge.

In the process of evolving WSN technology to IoT, emerging needs have been new challenges in the technology world. The main requirements in IoT technology are low power and low cost. These requirements make it difficult to meet standards such as heterogeneous device management, the communication of these devices with each other, security and privacy for IoT. Despite the growing number of IoT protocols and applications, there is not a global infrastructure. As is known, protocols such as Bluetooth Low Energy (BLE), ZigBee and 802.15.4 are the most preferred for IoT, and many mobile devices on the market today support these protocols. One of the reasons that such protocols are not standardized in IoT and cannot reach a global communication protocol is that the IPv6 packet structure is not small enough to fit 802.15.4 and similar BLE packets. The Ipv6 standard specifies the Maximum Transmission Unit (MTU) length as 1280 bytes, but the maximum frame size of the Physical Layer (PHY) of 802.15.4 is 127 bytes. Although it is stated that the optional L2CAP (Logical Link Control and Adaptation Protocol) packet fragmentation of the BLE supports this operation, it does not clear the issue of low energy and low cost that the 802.15.4 adopts [31]. At present, it is not possible to use Ipv6 structure in unique addressing principles of devices using these protocols, so IoT's internet access is performed by gateway devices or collecting sensor data such as MQTT(Message Queue Telemetry Transport), CoAP(Constrained Application Protocol), REST(Representational State Transfer), XML (Extensible Markup Language) Web services to cloud computing environments. However, wireless sensor node technologies have IPv6 applications on nodes that are typically used for Wi-Fi communication and have minimalist operating systems such as TinyOs, Contiki or RIOT OS. These operating systems have IPRoute and other protocol specific features on their own. Although studies have been conducted to transport IPv6 packets over BLE and significant values have been obtained in terms of performance [19] there has not yet been a formal improvements in the Bluetooth side. In December 2016, the Bluetooth 5 [1] version went through changes in symbol rate and increased range by improving error correction capabilities. Because of the increasing communication needs of all kinds of industrial production, healthcare, finance, energy and transport sectors and smart cities in the near future, the IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) [28] protocol, which has the IPv6 protocol, has been developed. Despite the existing barriers of 802.15.4, important step was taken for IoT with data compression methods. However, in most 6LowPAN applications, ZigBee and other 802.15.4 based radio frequency protocols are preferred and there is no improvement for BLE technology yet.

The rest parts of the paper are organized as follows: the technological background of BLE and virtualization are introduced in Section 2; related works are introduced in Section 3; protoype setup and experimental results shown are in Section 4; in section 5 we analyzed the performance of the prototype system.

2 MATERIALS AND METHODS

2.1 Bluetooth Low Energy (BLE)

Bluetooth low energy (formerly known as Bluetooth Smart) has been developed at middle of 2000's in version 4.0 of the Bluetooth Standard [20]. BLE operates in the unlicensed 2.4 GHz Industrial Scientific Medical (ISM) band. BLE divides the ISM band into 40 channels with 2 MHz spacing between center frequencies. Three predefined BLE advertisement channels (37, 38 and 39) operate as "advertising channels". While BLE technology was being developed, it used many features of classical Bluetooth technology. Power consumption and data transmission have been reduced by going through some restrictions in the host and physical layers of classical

Bluetooth. Reducing average energy consumption has enabled these devices to be activated for a much longer time. The reduction of the energy during data transmission means that the data size is directly reduced. Bluetooth low energy packets could be maximum 31 bytes long but this size in Bluetooth classical connection is up to 65535 bytes. In classical Bluetooth technology, when the data transmission delay time is 400 ms, this time in BLE is 3 ms [18].

There are two types of data packets used in BLE. These are advertisement and data packets. While advertisement packets are used by devices to discover, connect, and publish data, data packets are used to carry information on an established connection. Advertisement packets are used by broadcaster and observer devices, while data packets are used by master and slave devices. These packets are separated from each other by the frequency channel they are using. General structure of BLE packet is shown in Fig. 1.

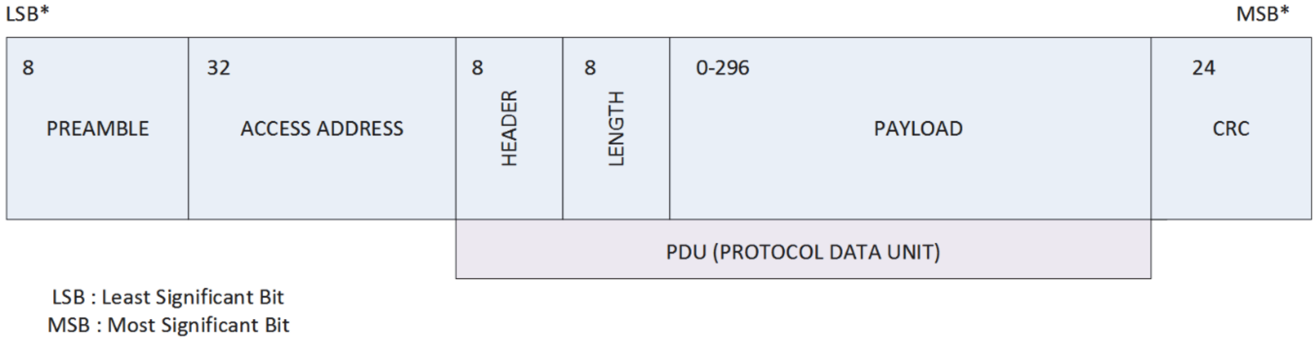


Figure 1 General BLE packet structure [1]

The content of the header field depends on whether it is an advertisement or a data packet. In the advertisement packet, the header field contains the sign bit that specifies the type of packet. This sign bit defines whether the packet has a public or random access address. The type of advertisement packet is also defined in the 4-bit section of the header field. There are seven types of advertisement packets listed in Tab. 1.

Table 1 Type of BLE advertisement packets.

Advertisement Packet Type	Value	Description
ADV_IND	0000	Connectable.
ADV_DIRECT_IND	0001	Connectable.
ADV_SCAN_IND	0110	Scannable
ADV_NONCONN_IND	0010	Not Connectable.
SCAN_RSP	0100	Scan Response.
SCAN_REQ	0011	Scan Request.
CONNECT_REQ	0101	Connection Request.

In this study ADV_IND and SCAN_RSP type advertisement packets are used.

2.2 Virtualization

Virtualization is generally defined as an abstraction layer between applications running on the operating system and hardware [12]. The goal of virtualization is to provide an integrated platform for users and applications by bringing together disparate and independent resources.

Virtualization types can be examined under five different headings:

- Server virtualization
- Desktop virtualization
- Application virtualization
- Storage virtualization
- Wireless sensor network virtualization.

Virtualization and network virtualization technologies have radically changed the way of information all over the world because of the increased needs for share and use of resources at the enterprise level, disaster recovery and cloud computing. At present, the number of virtual machines has already exceeded the number of physical servers [9]. Major service providers such as Google and Yahoo serve billions of people on the internet with virtual services. It would not have been possible for all these activities to be realized without virtual networks or virtual servers. All of these technologies are the basis of communication in today's world making it impossible to imagine a non-virtual server and network in terms of mobility and scalability.

2.3 Virtual Private Networks (VPNs) to Software-Based Networking (SDN)

Network virtualization can be defined as an instrument that is used primarily to provide isolation within an existing physical network infrastructure [14]. This structure, created between physical resources and users, creates an illusion that a single physical resource is being used as multiple and different networks. The first virtualization technologies used in computer networks started with virtual channels used in X.25 communication [24]. Later, the widespread adoption of virtual local area networks (VLANs) and similarly virtual private networks (VPNs) enabled the establishment and management of new networks for specific purposes, while preserving the existing security requirements of the used physical network structures.

There are basically two technologies in network virtualization. These are network device and link virtualizations [34]. Virtualization of network devices includes software or hardware based NIC(Network Interface Card) virtualization and network switch / router virtualization. NIC virtualization is one of the cornerstones of this technology and is offered in the solutions of operating system virtualization providers such as VMWare [10], Xen [11], Microsoft [3] and Oracle[6]. In addition, virtual switches are one of the leading components of today's network virtualization technology. OpenvSwitch [4] is the most preferred software-based network switch in recent times due to its performance and reliability. Apart from that, there are customized Linux and Berkeley Software Distribution (BSD) distributions such as Alpine, VxWorks which are used as network switch/router.

Link virtualization is defined as the virtualization of layers such as data plane / data path of network functions or multiplexing of transmission channel with time, frequency or code based methods in the network. Data plane / data path virtualization has led to the emergence of OpenFlow technology developed at Stanford University and the development of software-based networking technology (SDN) [30]. Although OpenFlow itself does not provide a fully virtualized environment, OpenFlow controllers such as FlowVisor can be considered virtualization because it creates an abstraction between the data path and control path layers of network switches [33].

2.4 OpenvSwitch

The network in the virtual environment is typically created in a management domain of a simple L2network switch or in a hypervisor. In such an environment it is preferred that virtual machines are connected to virtual network interfaces rather than direct access to Network Interface Cards (NICs). Virtual switches used in such networks provide a link between physical and virtual interfaces. At the same time, virtual network switches also control data traffic between virtual interfaces on the same physical host. Unlike physical switches, which are one but not system-to-network, virtual switches are in the main system that uses it, and they are entirely software. This removes the rigid structure caused by the hardware, thereby allowing the creation of distinctive routing functions and easily improved designs.

One of the most basic components of virtual networks, OpenvSwitch is now available in large virtualization systems such as VMWare, Xen, Kernel-based Virtual Machine (KVM) and Docker, as well as in open source software under the Apache 2.0 license. In terms of flexibility, reliability and performance, this software, which is the size of hardware based switches, [32] is quite capable of setting up and managing virtual networks. Unlike hardware-based network switches, the OpenvSwitch without a 24-48 port limit can have a number of ports that are allowed by the resources its own. These features have made it a very preferred product especially in test beds or academic network researches.

2.4 Container-Based Virtualization

There are a wide variety of server virtualization technologies that evolve in parallel with network virtualization technology, but they can basically be grouped into two main groups [13], hypervisor and container-based virtualization. Instances of hypervisor-based virtualization technologies are Xen, KVM. Hypervisor-based virtualization provides a multitasking environment by acting as a bridge between host and virtual operating systems running on the host and called guest. Hypervisor virtualization has two methodologies: system emulation and paravirtualization without hardware virtualization.

Container-based virtualization is an operating system-level virtualization technology. It can be assumed that BSD chroot applications began in the 1980's [29]. In container-based virtualization, the kernel is implemented with ABI(Application Binary Interface) virtualization technology and hypervisors are not used [35]. The basic principle of container-based virtualization is to allocate resources and perform isolation in the same operating system without the need for extra hardware. Operating systems such as OpenVZ [5], Solaris 10 [8] use this method. Over the years, container-based virtualization technology has become very popular and has become part of the official Linux Kernel.

Linux Network Namespace (NetNS) which is the one of the container-based virtualization is designed to meet the network isolation requirement within the Linux operating system. NetNS copies the network stack structure into a special set of network resources assigned to one or more processes. This resource set includes devices, IP address, port space, route table, and more. This set also has a shell process [16]. With these features NetNS can also be configured as an ethernet bridge or router at the same time. Since NetNS is a solution focused solely on network isolation, it makes the NetNS simpler than OpenVZ and VServer. Although it is known that there are some conflicts in some operating systems, it is a preferred technology in terms of container-based virtualization [15].

3 RELATED WORKS

Most network virtualization technologies have been developed for wired networks. However, researches are also underway for wireless networks, including mobile cellular and sensor networks. In this context, wireless virtualization techniques are evaluated within the context of link virtualization [34].

Wireless sensor networks have many sensor nodes with limited resources and are adapted to work with specific applications. However, there are physical difficulties in wireless virtualization technologies because they are sensitive to data latency [22]. SDN technology has a significant potential in the future as a good alternative to wireless virtualization solutions. With OpenFlow, management of network structure and topology can be made more flexible. This approach facilitates researchers to overcome difficulties for wireless networks [26]. For instance, SD (Software Defined)-WSN [27], is used as a structure that separates the control plane and the data plane from each other based on the Sensor OpenFlow (SOF) technology. In this architecture, sensor nodes transmit flow-based packets while the controller centralizes all network functions with QoS (Quality Of Service) support. Other virtualization solution is IMPERIA [21]. IMPERIA uses cluster structure for large scale wireless sensor networks. At the center of each WSN cluster is a gateway node that manages sensor nodes. These Gateway nodes are also managed by a global controller.

Similarly, the SDN-TAP (Traffic Aware Protocol) has been developed for congestion management and automatic topology generation in wireless networks with using software-based network technology to optimize communication among wireless sensor networks [17]. On the other hand, in the VISE (Virtualize sensing environment) [23] the control plane and the data planes are separated from each other and virtualization and cloud computing technologies are used together.

As a result of such researches, the concept of Virtual Sensor Network (VSN) [26] has emerged in recent years and the conceptual backgrounds of how to utilize existing resources in the most efficient way through the virtualization of functions of traditional sensor networks are discussed.

4 SETUP

In this study, Raspberry Pi3 [7] was used as Container-based virtualization host. This model of Raspberry Pi3 has integrated Bluetooth 4.1 data and BLE support. With its hardware features, RaspberryPi3 is capable of meeting the test environment in this work. In addition, 45 indoor positioning BLE sensors are used for virtualization. On these devices, CC2541 [2] SimpleLink Bluetooth low energy and proprietary wireless micro controller and 3 Volt coin cell battery are combined (Fig. 2).

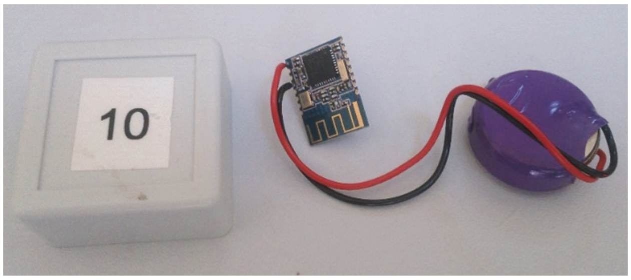


Figure 2 BLE Indoor positioning beacon

They are used to advertise Received Signal Strength Indicator (RSSI) values for various applications. The CC2541 microchip is advertise building ID, room ID etc. data order to be an example in this study. Sample packet structure is shown Fig. 3.

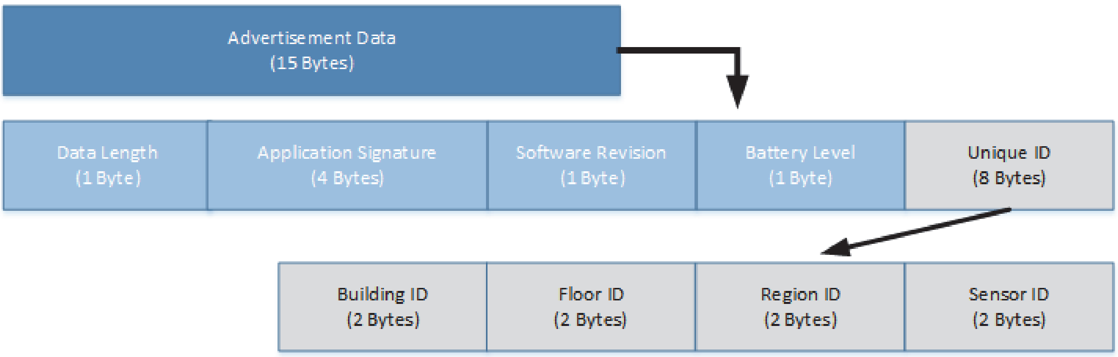


Figure 3 Example advertisement data

The Raspbian Linux (Core version 4.4.50) operating system on Raspberry Pi3 uses the prototype Python application with pyBluez, the Python port of the Bluez Protocol stack, and the basic Python libraries, which transform each BLE node into a NetNS with container based virtualization. The prototype Python application scans and receives ADV_IND or ADV_SCAN_RSP information from sensor devices then adds them an unique list of devices. While adding device to list, it creates a container that uses OpenvSwitch and NetNS commands. The container is created in Linux and each corresponding sensor device receives a real IP address of the campus via a DHCP server in the network where the study is being done. At the same time, these containers run a simple REST server application on them. The model created by the virtualizer application can be seen in Fig. 4.

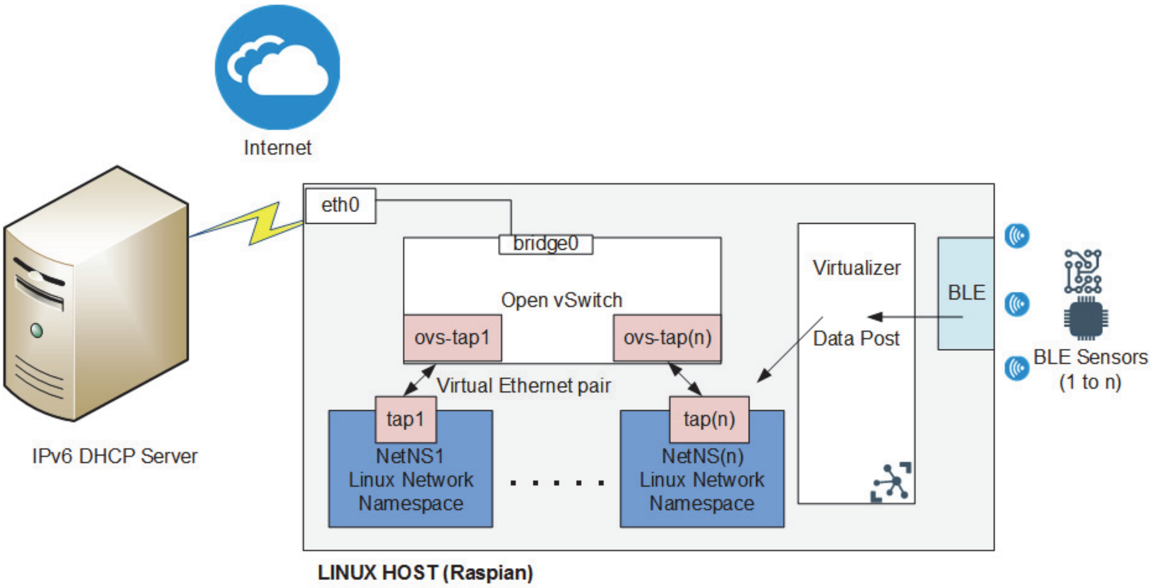


Figure 4 Prototype system and created model. Although the number *n*, which indicates the number of sensors, has no limit, in theory, it can be limited to 64 in terms of communication performance

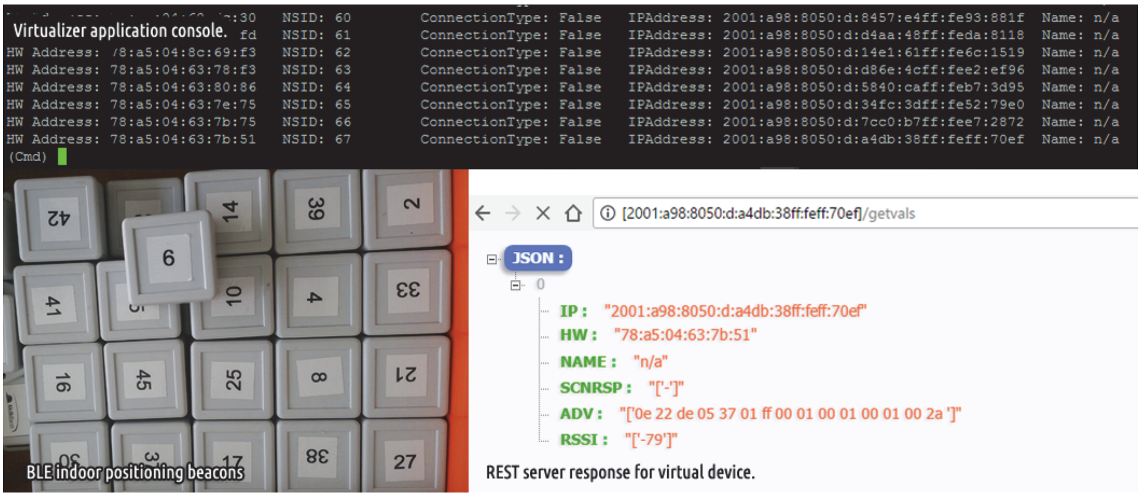
The pseudocode of the functions used to find BLE enabled devices and process the data is shown in Fig. 5.

In this work, BLE data obtained by the virtualizer application stored into devices object. For each item in this list own advertisement data send to server running in the corresponding virtual namespace instantaneously. In the experiment applied in this study, a sample screen produced by the virtualization software for 45 beacon devices and the response of the REST server representing the virtual devices is shown in Fig. 6.

```
function ScanDevices();
Input : none
Output : none
do while true
    Parse devices;
    if device not found in deviceList then
        add device to deviceList;
        StartVirtualDevice();
    else
        find deviceItem in deviceList;
      if advertimesentType = Scan Response  or Indirect Advertisement then
          post raw data;
      end
    end
```

```
function StartVirtualDevice();
Input : none
Output : none
for deviceList
if deviceItem has no Ipv6 Address then
    find network namespace count;
    add new network namespace;
    add tap device , open vswitch virtual pair;
    add ovs pair to bridge port;
    set namespace device tap device;
    device up;
    get ipv6 from dhcp server;
    set ipv6 address in device list;
    start web server for network namespace;
end
```

Figure 5 Pseudocode of the virtualizer application


Virtualizer application console.
42
14
39
2
6
41
5
10
4
33
16
45
25
8
21
30
3
17
38
27
BLE indoor positioning beacons
JSON :
0
REST server response for virtual device.

```
                             :30   NSID: 60       ConnectionType: False    IPAddress: 2001:a98:8050:d:8457:e4ff:fe93:881f   Name: n/a
                              fd   NSID: 61       ConnectionType: False    IPAddress: 2001:a98:8050:d:d4aa:48ff:feda:8118   Name: n/a
HW Address: /8:a5:04:8c:69:f3   NSID: 62       ConnectionType: False    IPAddress: 2001:a98:8050:d:14e1:61ff:fe6c:1519   Name: n/a
HW Address: 78:a5:04:63:78:f3   NSID: 63       ConnectionType: False    IPAddress: 2001:a98:8050:d:d86e:4cff:fee2:ef96   Name: n/a
HW Address: 78:a5:04:63:80:86   NSID: 64       ConnectionType: False    IPAddress: 2001:a98:8050:d:5840:caff:feb7:3d95   Name: n/a
HW Address: 78:a5:04:63:7e:75   NSID: 65       ConnectionType: False    IPAddress: 2001:a98:8050:d:34fc:3dff:fe52:79e0   Name: n/a
HW Address: 78:a5:04:63:7b:75   NSID: 66       ConnectionType: False    IPAddress: 2001:a98:8050:d:7cc0:b7ff:fee7:2872   Name: n/a
HW Address: 78:a5:04:63:7b:51   NSID: 67       ConnectionType: False    IPAddress: 2001:a98:8050:d:a4db:38ff:feff:70ef   Name: n/a
(Cmd)
```

[2001:a98:8050:d:a4db:38ff:feff:70ef]/getvals

```
IP :  "2001:a98:8050:d:a4db:38ff:feff:70ef"
HW :  "78:a5:04:63:7b:51"
NAME :  "n/a"
SCNRSP :  "['-']"
ADV :  "['0e 22 de 05 37 01 ff 00 01 00 01 00 01 00 2a ']"
RSSI :  "['-79']"
```

Figure 6 Indoor positioning beacons and virtual servers.

5 RESULTS

The average duration of Namespace creation on the platform used for 45 BLE Indoor positioning beacon devices measured is 2,504 seconds. This duration includes the sum of the DHCP response/request times and container creation for the BLE devices that are found within the communication range of the virtualization software. It is considered that this value is acceptable level for the application, such as Raspberry Pi3 platform.

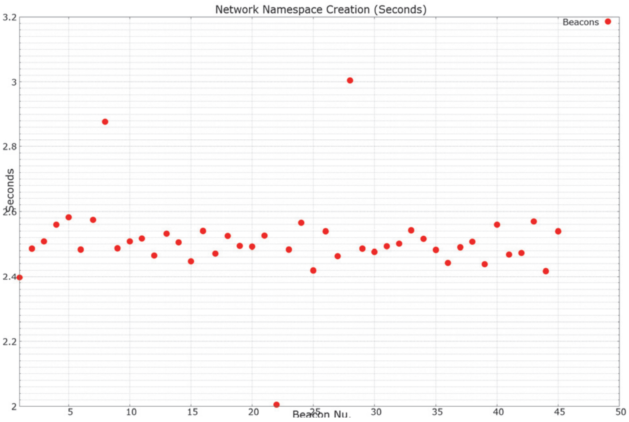


Figure 7 Network Namespace creation with assigned IP address

The details of the namespace creation times for devices are depicted in Fig. 7.

The REST server running on namespaces created have been tested for performance over concurrent 50 users on the fly. As a result, it has been calculated that these servers are able to respond for an average of 1180 ms for requests. Globally it has been seen that 400 users' requests can be met on average.

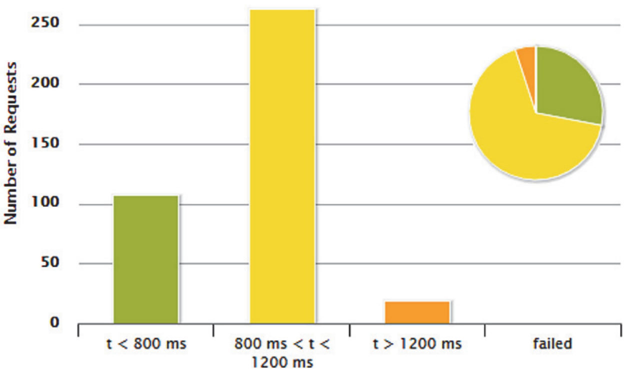


Figure 8 Number of requests and durations

The results obtained are shown in the graphs in Fig. 8 and 9.

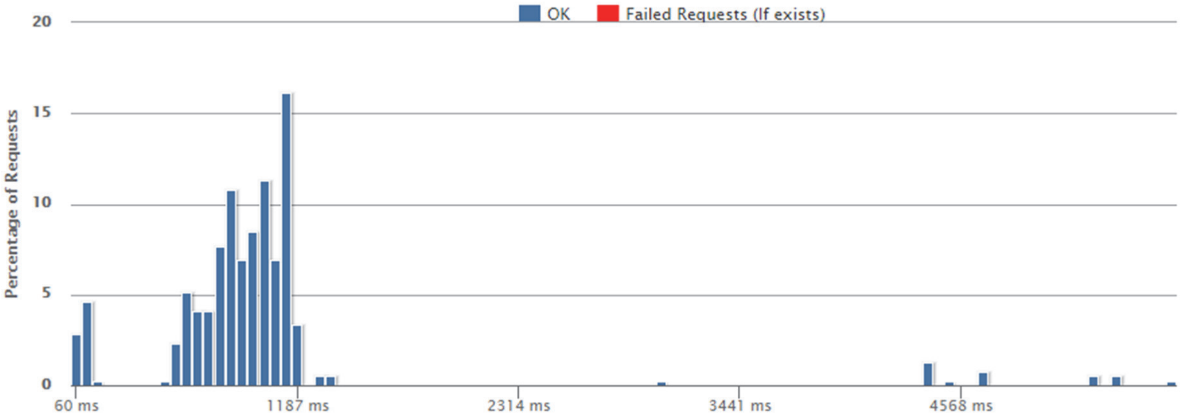


Figure 9 Response time distribution in milliseconds

6 CONCLUSIONS

It is envisaged in the near future that the smart devices used in energy, finance, agriculture, logistics and any kind of production will be accessible to the internet and communication with each other. Currently, these communication technologies have been used in some aspects. One of the important components of the IoT paradigm is that the sensor and actuator devices used in e any production fields as well as the regular devices used on a daily basis will have access to the internet. The basic requirements of this access are low power consumption and low cost as well as IPv6 technology. But, it seems that the technologies for realizing these three components at the same time are still in trial stage. Ipv6 enabled communication conditions do not seem to provide low power consumption in wireless sensor networks and IoT applications. Except Wi-Fi solutions at the present, Bluetooth low energy, ZigBee and other technologies today have obstacles to IPv6 communication and there are currently no practical solutions to be used effectively. In this study, a prototype method and software for providing IPv6 communication by using container-based virtualization method for low power consumption sensor devices that BLE enabled and do not have micro operating system have been developed and tested. Our prototype seems to be a useful approach in terms of the number of instant users and responsiveness to web requests except for critical tasks. Container-based virtualization is expected to provide developers and manufacturers with an opportunity for future security concerns in IoT applications as the features of the Linux operating system.

Another advantage is that the services and characteristics of the actuator devices will be able to be controlled more flexibly thanks to the advantages provided by virtualization. This generalization can also be a promising approach to establishing a standard for the functionality of many different IoT-enabled sensors and actuators.

7 REFERENCES

[1] See https://www.bluetooth.com/specifications/adopted-specifications

[2] See http://www.ti.com/product/CC2541

[3] See https://www.microsoft.com/windowsserversystem/virtualserver/

[4] See http://openvswitch.org/

[5] See https://openvz.org/

[6] See https://www.virtualbox.org/

[7] See https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[8] See https://www.oracle.com/solaris/solaris10/index.html

[9] See https://news.microsoft.com/2013/07/08/steve-ballmer-worldwide-partner-conference-2013-keynote/

[10] See http://www.vmware.com

[11] See http://www.xen.org

[12] Adams, K. & Agesen, O. (2006). A comparison of software and hardware techniques for x86 virtualization. *ACM SIGOPS Operating Systems Review, 40*(5), 2-13. https://doi.org/10.1145/1168917.1168860

[13] Bardhi, B., Claudi, A., Spalazzi, L., Taccari, G., & Taccari, L. (2016). Chapter 6 - Virtualization on embedded boards as enabling technology for the Cloud of Things A2 - Buyya, Rajkumar. *Internet of Things*, 103-124. https://doi.org/10.1016/B978-0-12-805395-9.00006-X

[14] Carapinha, J. & Jiménez, J. (2009). Network virtualization: a view from the bottom. *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, Barcelona, Spain. https://doi.org/10.1145/1592648.1592660

[15] Chen, M. & Nakao, A. (2010). Feather-Weight Network Namespace Isolation Based on User-Specific Addressing and Routing in Commodity OS. *International Conference on Testbeds and Research Infrastructures*, 53-68. https://doi.org/10.1007/978-3-642-17851-1_4

[16] Ferrari, M. (2010). NEPI: An experiment plane for experimentation test beds. *End of studies report for MASTER II IFI (Informatique: Fondements et Ingénierie) - Parcus UBINET - Ubiquitous Network Computer - POLYTECH Nice Sophia*, 2010-09-30.

[17] Fotouhi, H., Vahabi, M., Ray, A., & Björkman, M. (2016). SDN-TAP: An SDN-based traffic aware protocol for wireless sensor networks. *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, 1-6. https://doi.org/10.1109/HealthCom.2016.7749527

[18] Gupta, N. (2013). *Inside Bluetooth Low Energy*. Artech House.

[19] Haolin, W., Minjun, X., Jia, L., & Canfeng, C. (2013). Transmitting IPv6 packets over Bluetooth low energy based on BlueZ. *2013 15th International Conference on Advanced Communications Technology (ICACT)*, 72-77.

[20] Honkanen, M., Lappetelainen, A., & Kivekas, K. (2004). Low end extension for Bluetooth. *Proceedings. 2004 IEEE Radio and Wireless Conference (IEEE Cat. No.04TH8746)*, 199-202.

[21] Hunkeler, U., Lombriser, C., Truong, H. L., & Weiss, B. (2013). A case for centrally controlled wireless sensor networks. *Computer Networks, 57*(6), 1425-1442. https://doi.org/10.1016/j.comnet.2012.12.019

[22] Irwin, D., Sharma, N., Shenoy, P., & Zink, M. (2010). Towards a virtualized sensing environment. *International Conference on Testbeds and Research Infrastructures*, 133-142. https://doi.org/10.1007/978-3-642-17851-1_10

[23] Islam, M. M., Hassan, M. M., Lee, G.-W., & Huh, E.-N. (2012). A survey on virtualization of wireless sensor networks. *Sensors, 12*(2), 2175-2207. https://doi.org/10.3390/s120202175

[24] Jain, R. & Paul, S. (2013). Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine, 51*(11), 24-31. https://doi.org/10.1109/MCOM.2013.6658648

[25] James, R. (2014). The Internet of Things: A study in Hype, Reality, Disruption, and Growth. *Raymond James US Research, Technology & Communications*, *Industry Report*.

[26] Jayasumana, A. P., Han, Q., & Illangasekare, T. H. (2007). Virtual sensor networks-a resource efficient approach for concurrent applications. *Information Technology, ITNG'07. Fourth International Conference on*, 111-115. https://doi.org/10.1109/ITNG.2007.206

[27] Luo, T., Tan, H.-P., & Quek, T. Q. (2012). Sensor OpenFlow: Enabling software-defined wireless sensor networks. *IEEE Communications Letters, 16*(11), 1896-1899. https://doi.org/10.1109/LCOMM.2012.092812.121712

[28] Ma, X. & Luo, W. (2008). The Analysis of 6LowPAN Technology. *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, 1*, 963-966. https://doi.org/10.1109/PACIIA.2008.72

[29] Matthias, K. & Kane, S. (2015). *Docker: Up and Running*. Sebastopol: O'Reilly Media, Inc.

[30] McKeown, N. et al. (2008). OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review, 38*(2), 69-74. https://doi.org/10.1145/1355734.1355746

[31] Mikhaylov, K., Plevritakis, N., & Tervonen, J. (2013). Performance Analysis and Comparison of Bluetooth Low Energy with IEEE 802.15.4 and SimpliciTI. *Journal of Sensor and Actuator Networks, 2*(3), 589. https://doi.org/10.3390/jsan2030589

[32] Pfaff, B. et al. (2015). The design and implementation of open vSwitch. *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, Oakland, CA.

[33] Sherwood, R. et al. (2009). Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, 1-13.

[34] Wang, A., Iyer, M., Dutta, R., Rouskas, G. N., & Baldine, I. (2013). Network Virtualization: Technologies, Perspectives, and Frontiers. *Journal of Lightwave Technology, 31*(4), 523-537. https://doi.org/10.1109/JLT.2012.2213796

[35] Xavier, M. G., Neves, M. V., Rossi, F. D., Ferreto, T. C., Lange, T., & De Rose, C. A. (2013). Performance evaluation of container-based virtualization for high performance computing environments. *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, 233-240. https://doi.org/10.1109/PDP.2013.41

Contact information:

Deniz TAŞKIN, PhD
Trakya University, Computer Engineering Department,
Edirne 22030, Turkey
E-mail: deniztaskin@trakya.edu.tr

Cem TAŞKIN, PhD
Trakya University, Computer Engineering Department,
Edirne 22030, Turkey
E-mail: cemtaskin@trakya.edu.tr

Selçuk YAZAR
(Corresponding author)
Trakya University, Computer Engineering Department,
Edirne 22030, Turkey
E-mail: selcukyazar@trakya.edu.tr