WeBLE: A BLE-based system compliant with WoT architecture

Yi-Hsiu Chiu∗, Chun-Feng Liao∗†
∗Department of Computer Science
†Program in Digital Content and Technologies
National Chengchi University, Taipei, Taiwan

***Abstract*—The W3C organization has recently continued to propose recommendation for WoT (Web of Things). In order for BLE, which has a high market share in IoT application, to comply with the WoT recommendation, this paper propose a proprietary application-level gateway and WeBLE (Web of BLE) device that allow clients to treat BLE devices as WoT node and increase the interoperability of BLE devices.**

Keywords—Internet of Things, Web of Things, Bluetooth Low Energy, CoAP

I. INTRODUCTION

The advancement of IoT (Internet of Things) technologies is maturing the implementation of Smart Environments and their application in daily life. Recently, W3C has proposed WoT (Web of Things) recommendations [1]. In WoT architecture, any operation should be based on the web architecture using RESTful API and devices should be described by WoT Thing Description. WoT improves the interoperability and usability of IoT. However, due to power consumption and computing power considerations. IoT devices typically use a different protocol stack than Internet devices. This makes it difficult for IoT devices to communicate in WoT architecture directly. While 6LoWPAN technology provides IPv6 addressing capabilities for IoT devices, enabling IoT devices to run application layer protocols that are compliant with WoT architecture. For example, Mohiuddin et al. employed SSLP (Simple Service Location Protocol) and CoAP (Constrained Application Protocol) on IEEE 802.15.4 and 6LoWPAN-based nodes [2]. Among them, CoAP is a RESTful protocol that conforms to WoT architecture. But 6LoWPAN is only available for IEEE 802.15.4. For non-IEEE 802.15.4-based protocols, such as BLE (Bluetooth Low Energy), although it also has 6LoWPAN for BLE, it is not common in practice and the system was reported to be unstable by Chawathawornchaoen et al. [3]. In order to make non-IEEE 802.15.4-based protocol compliant with WoT. This paper uses BLE as an example and propose a proprietary gateway and WeBLE (Web of BLE) device to solve the above problem from application-level.

II. SYSTEM OVERVIEW

The system overview in this paper is shown in Figure 1. On the right side of the figure, we implemented WeBLE (Web of BLE) devices which are based on BLE. And in the middle, there is a gateway responsible for registering the surrounding WeBLE services and providing clients to discover or request WeBLE services using CoAP from the internet. For clients (left in the figure), they won't feel like they are communicating with the gateway but interacting with the WoT node directly. To sum up, our system empowers BLE to comply with WoT architecture through the design of the gateway and WeBLE device. And in the next chapter, we shall explain the design details of WeBLE devices and the gateway with a complete system flow (service registration, service discovery and request routing).

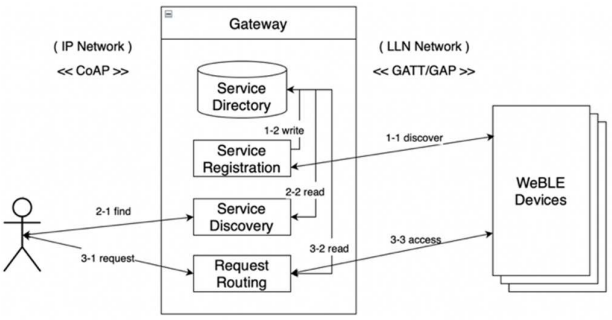

Fig. 1. System overview

III. SYSTEM DESIGN

A. Service Registration

As with most IoT networks, our system uses a service directory to manage services in the network to reduce device wake-up time. The specific way is that the gateway uses GAP and GATT to discover WeBLE devices and their services and write into the service directory periodically. But how does the gateway identify whether the device is a WeBLE device or a normal BLE device? For this purpose, our system refers to IPSP (Internet Protocol Service Profile) in RFC 7668 [4]. We propose WeBLE Façade Service to enable device to declare itself as WeBLE device. Specifically, WeBLE Façade service is a specific BLE service UUID, device can carry this UUID in the GAP broadcast packet to let observers know that it provides this service. In this way, the device can declare itself as a WeBLE.

B. Service Discovery

With service directory and service registration, the gateway can manage services in the network. And for clients to know what services are provided in this network, our system provides service discovery. As mentioned, our system is compliant with WoT architecture for clients, so for the client side, we use CoAP to communicate. And CoAP usually uses well-known URI to implement service discovery, so our system uses well known URI as the service discovery interface. Clients can send a request to the gateway that matches well-known URI format. Upon receipt of the request, the gateway will query the service directory for the matching service and send it back to the clients.

C. Request Routing

To allow clients to interact with WeBLE services using WoT style. The gateway in our system needs a mechanism to convert the CoAP request to a BLE GATT request and then route it to the target device. According to the format of the request, the mechanism needs to do the following conversions:

$$Req_{CoAP}(IP, Methods, URL) \Rightarrow Req_{BLE}(BLEAddr, Properties, UUID) \quad (1)$$

In equation (1), we can see that the conversion can be divided into three parts: 1) IP to BLE address conversion, 2) Request methods to characteristic properties conversion, and 3) URL to UUID conversion. Each part is explained below:

- **IP to BLE address conversion:** For the conversion of IP to BLE addresses, we refer to Stateless Address Autoconfiguration in RFC 7668 [4], which allows a 48-bit BLE address to be converted to a 128-bit ipv6 stateless address. The specific conversion is to insert the 0xFFFE in the middle of the BLE address to form a 64-bit IID (Interface Identifier), and then add the IPv6 local address prefix to form a 128-bit ipv6 address.
- **Request methods to characteristic properties conversion:** As mentioned, devices in our system (i.e. WeBLE) are based on BLE, and BLE GATT characteristic has three properties for clients to interact with the device, which are read, write and notify. To achieve our research objectives, we need to correspond these properties with CoAP request methods. Table 1 demonstrates correspondence of properties. Read and write can be corresponded to GET and PUT intuitively. Notify provides server push capability for BLE characteristic, which cannot be corresponded with ordinary request methods directly, but needs to achieve through the observe flag defined by RFC 7641 [5].

TABLE I. CORRESPONDENCE OF PROPERTIES

CoAP request methods	*BLE characteristic properties*
GET	READ
PUT	WRITE
GET + Observe flag	NOTIFY

- **URL to UUID conversion:** CoAP URLs are string-based, and most of them have semantics. However, in BLE GATT, services and characteristics are represented as hexadecimal-based UUIDs. In order to convert it, we need to query the UUID's feature description from 16-bit UUID numbers document and use it as the name of that UUID [6]. For example, in the document, the UUID {0x180f, 0x2a19} represents the "Battery" service and "Battery Level" characteristic. So we can use the URL {Battery/BatteryLevel} to denote the UUID {0x180f, 0x2a19}. However, self-defined UUIDs are not available from the document, so we define a specific characteristic and descriptor to name the service and characteristic UUID, called Service Name Characteristic and Characteristic Name Descriptor. Specific approach as shown in Figure 2, add Service Name Characteristic and Characteristic Name Descriptor under target service and characteristic then fill in the name in value. In this way, we can get the name of the target by reading its Service Name Characteristic or Characteristic Name Descriptor.

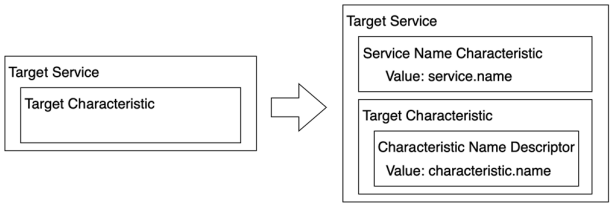

Fig. 2. Conversion mechanism for self-defined UUID

IV. EVALUATION

This section reports and discuss the evaluating result of our system. Our system uses a Raspberry Pi to simulate the gateway. Raspberry Pi has multiple network interfaces (e.g., Ethernet, WLAN and Bluetooth) that allow it to communicate with clients and WeBLE devices over Ethernet and BLE. For the WeBLE device, we simulated it with NodeMCU 32s, which provides BLE interface and is suitable for installation in IoT devices. And for the client, we simulate it with a normal PC computer.

For the evaluation of our system, we designed an experiment to compare the request/response time of our system with BLE network. The result is shown in figure 3. From the diagram, we can see that the time difference is not significant, for each property, the additional time spent by WeBLE is less than 5%. However, compared to BLE, WeBLE allows the client to treat BLE-based devices as web nodes, which increases the interoperability of BLE.

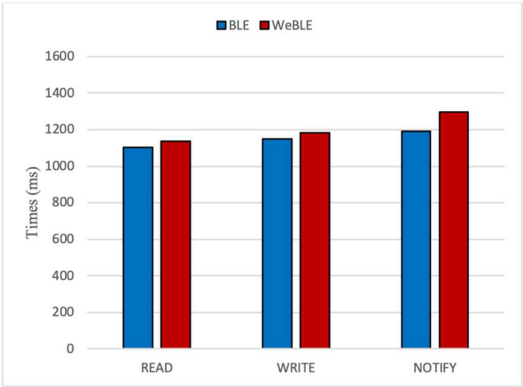

Fig. 3. Request/Response time comparison between WeBLE and BLE

REFERENCES

[1] Kovatsch, M., Matsukura, R., Lagally, M., Kawaguchi, T., Toumura, K., and Kajimoto, K. (2020). Web of things (wot) architecture, recommendation. Technical report, World Wide Web Consortium (W3C).

[2] Mohiuddin, J., Bhadram, V., Palli, S., and Koshy, S. S. (2014). 6LoWPAN based service discovery and RESTful web accessibility for Internet of Things. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 24-30). IEEE.

[3] Chawathawoncharoen, V., Visoottiviseth, V., and Takano, R. (2015). Feasibility evaluation of 6LoWPAN over Bluetooth low energy. *arXiv preprint arXiv:1509.06991*.

[4] Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., & Gomez, C. (2015). *Ipv6 over bluetooth (r) low energy* (No. rfc7668).

[5] Hartke, K. (2015). Observing resources in the constrained application protocol (CoAP) (No. rfc7641).

[6] BLE-UUID (2021). 16-bit uuid numbers document, rev. 2021-12-03. Technical report, Bluetooth SIG, Inc.