A BLE-based multi-gateway network infrastructure with handover support for mobile BLE peripherals

Mathias Baert
IDLab - Ghent University - imec
Technologiepark Zwijnaarde 126,
B-9052 Ghent, Belgium
Email: mathias.baert@UGent.be

Pieterjan Camerlynck
Televic Healthcare,
Leo Bekaertlaan 1,
8870 Izegem, Belgium
Email: p.camerlynck@televic.com

Pieter Crombez
Televic Healthcare,
Leo Bekaertlaan 1,
8870 Izegem, Belgium
Email: p.crombez@televic.com

Jeroen Hoebeke
IDLab - Ghent University - imec
Technologiepark Zwijnaarde 126,
B-9052 Ghent, Belgium
Email: jeroen.hoebeke@UGent.be

***Abstract*—Bluetooth Low Energy (BLE) is a popular technology within the Internet of Things. It allows low-power, star networks to be set up between a BLE gateway and multiple, power-constrained BLE devices. However, these networks tend to be static, not supporting BLE devices that can freely move around in an environment of multiple interconnected BLE gateways and perform handovers whenever necessary. This work proposes two alternative network architectures for mobile BLE peripherals. One leverages on IPv6 over BLE, whereas the other combines default BLE mechanisms with an additional custom controller. On top, we study in detail the handover mechanism that must be present in both architectures and compare the performance of both a passive and active handover approach. The passive handover approach can be set up without any extra implementation, but an active handover approach offers more proactive handover decisions and can provide a much lower handover latency. All proposed solutions have been implemented and validated on real hardware, showing the feasibility of having future infrastructures with support for mobile BLE devices.**

***Index Terms*—IoT, BLE, IP, Non-IP, handover, mobility**

I. Introduction

Today, WiFi networks have become ubiquitous in people's everyday life. They deliver continuous device connectivity even in the presence of mobility thanks to handovers between WiFi access points. With the expansion towards the Internet of Things (IoT), more and more use cases pop up that also want to benefit from such seamless connectivity. However, a lot of these use cases involve energy constrained devices (wearables, sensors, etc.) and thus require a low-power optimized approach. Therefore, several organizations developed standardized protocols more suited for the IoT [1]. A well-known and widespread offspring of this endeavor, is Bluetooth Low Energy (BLE) [2].

BLE can be employed to set up a connection between a central device and one or more peripherals. In typical use cases, a person is carrying both the central and the peripheral, such as a smartphone and a healthcare wearable, or needs to stay within the vicinity of either one of the two when carrying the other. Opposed to WiFi, it is not yet possible to freely move around with a peripheral and interact with a BLE-enabled network infrastructure [2]. Such functionality would enable use cases in healthcare, Industry 4.0 and other domains, where people carrying wearables can be continuously monitored by a BLE infrastructure present in the building.

Considering this gap, it is exactly the goal of this work to investigate the realization of a BLE-enabled infrastructure of IoT gateways (GW) and the possibility of seamless handovers with a mobile BLE peripheral within that network, thus maintaining a connection between a mobile BLE peripheral and an IoT gateway at all times. This mirrors the concept of handovers within a network of WiFi access points, but now optimized for power constrained devices. The GWs are all part of the same IP network, together with any existing application servers already present in that network. These servers can communicate with the peripheral via the IoT GWs (and vice versa). Two approaches are presented to realize such an architecture. Both of them offer **seamless handovers** within a network of BLE IoT GWs, provide **bidirectional communication** and maintain **end-to-end application semantics**.

The latter property addresses the *gateway problem* currently at large within the IoT [3] due to application specific connectivity between an IoT GW and an IoT device. As this practise hampers the growth potential of the IoT, current research tends to evolve towards application agnostic IoT connectivity solutions. This research embraces this evolution as well.

A key contribution of this work is the design, validation and in-depth discussion of two architectures capable of achieving seamless and application-agnostic handovers within a network of BLE-enabled GWs. On top, both a passive and active handover strategy is explored and the handover performance of both approaches is evaluated, exploiting observations made in [4] and [5] about the impact of several BLE parameters on latency and energy consumption during neighbor discovery.

The rest of the paper is organized as follows. Section II gives an introduction on BLE and enabling IPv6 over BLE. Section III describes related work on the different aspects involved in this research. Section IV presents the two architectural designs to enable the envisioned BLE-enabled networking infrastructure, followed by the two handover strategies that can be applied to both architectures in Section V. The implementation-specific details are given in Section VI followed by an evaluation in Section VII. Finally, some concluding remarks and possible next steps are given in Section VIII.

II. Primer on BLE and BLE over IPv6

A. *Bluetooth Low Energy*

BLE is a popular technology within the IoT. It was released in 2011 as part of the Classic Bluetooth specification and both technologies have coexisted and evolved independently since then. Today, support for these technologies exists in laptops, smartphones, tablets, smart watches, etc. BLE operates in the 2.4 GHz band, utilizing frequencies between 2402 and 2480 MHz. The used spectrum is divided into 40 channels, each employing a space of 2 MHz. These channels are divided into 3 primary advertisement channels and 37 connection oriented channels. BLE supports two different ways of communication: an advertising mode and a connection oriented mode.

Advertising mode. In this mode, a device can either be in an advertising or scanning state. A BLE advertiser broadcasts advertisement packets on the 3 advertisement channels sequentially, at a certain interval. A BLE scanner scans one of these advertisement channels, also shifting between them at a certain interval. The important parameters in this communication method are the *advertising interval* at which the BLE advertiser broadcasts, the *scan interval* at which the BLE scanner shifts between the advertisement channels and the *scan window* which indicates how long a scanner actually scans between the intervals. If the scan interval and the scan window are the same value, then the BLE scanner scans at a 100% duty cycle.

A BLE advertiser can broadcast different types of advertisements and a BLE scanner can be configured to listen for a specific type of advertisement as well. An advertisement can either be connectable or non-connectable (other types also exist, but are out of scope for this paper). If a BLE scanner is configured to listen for non-connectable advertisements, it is in the *scanning* state. Otherwise, if the BLE scanner is configured to listen for connectable advertisements, it is in the *initiating* state, meaning it has the intention of employing these advertisements to initiate a BLE connection with a device that is broadcasting connectable advertisements. In order to set up such a connection, the *initiator* sends a connection request back to the broadcasting device, using the advertisement channel where the connectable advertisement was received on. To be able to receive a connection request, the broadcasting device should be listening on that advertisement channel as well. So after transmitting a connectable advertisement on an advertisement channel, the BLE advertiser listens on that channel until the part of the advertising event for that advertisement channel, is over.

Connection oriented mode. In a BLE connection, one device acts as a BLE central and the other device acts as a BLE peripheral. Both of them are in a *connection* state. The devices communicate within connection events, at a certain interval and initiated by the central. A FHSS technique is used to hop between the 37 connection oriented channels. If no application layer communication is done at the higher level, the idle BLE connection is maintained by transmitting one empty packet each. A BLE central can be connected to multiple peripherals at once, by employing a TDMA access scheme. The important parameters in this mode are the *connection interval* at which frequency hopping occurs, the *slave latency* which can allow the peripheral to skip a certain amount of connection events to save power and the *supervision timeout* which indicates how long no communication can occur until a connection is considered lost. Whenever a BLE packet is received through the connection, the latter timeout is reset. On top of such a connection, a BLE peripheral and BLE central communicate based on predefined or custom BLE services. These services are used by the BLE peripheral to indicate to the BLE central what features it offers (e.g. the battery service is able to provide insight into the battery's performance and can be used to tweak its settings).

This work mainly employs the connection oriented mode within both solutions. The advertising mode is used to setup a BLE connection and to achieve an active handover approach (see Section V).

B. *IPv6 over BLE*

Also the enablement of end-to-end IPv6 connectivity on top of BLE has been considered in standardization, as presented in [6]. It exploits the existing BLE stack layers and adds a 6LoWPAN layer between the BLE layers and the IP layer, as shown in Figure 1. The BLE central acts as 6LoWPAN border router (6LBR) and one or more BLE peripherals as 6LoWPAN nodes (6LN), resulting in an isolated IPv6 over BLE subnet. The 6LoWPAN border router can also be connected to the Internet, ultimately allowing end-to-end IPv6 connectivity between a BLE peripheral and any application server on the Internet.

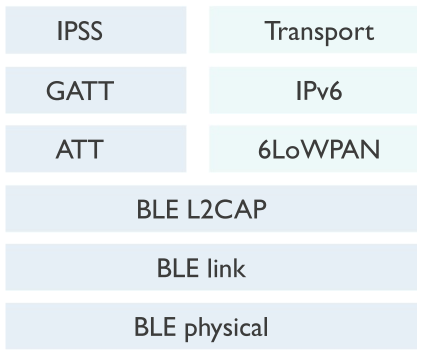


Fig. 1: The stack extension to enable IPv6 over BLE.

To establish an IPv6 enabled BLE connection between a 6LBR and a 6LN, several steps are involved. First, a native

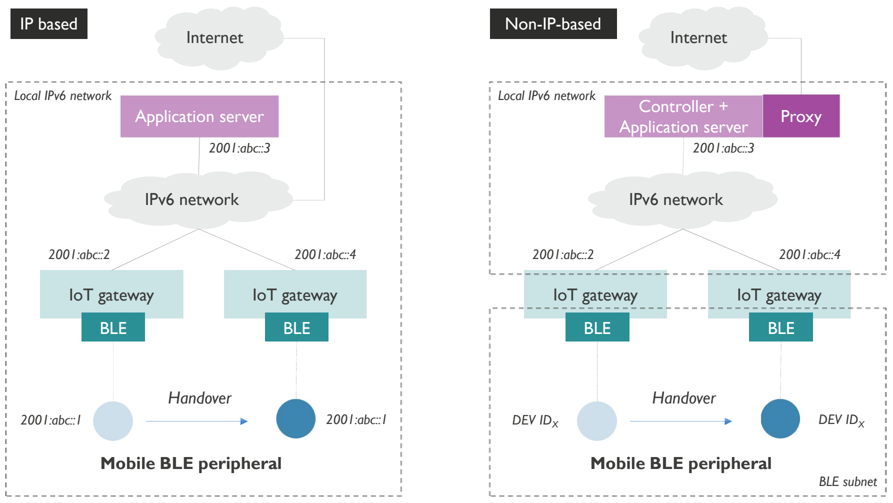


Fig. 2: The two proposed solutions based on an IP (on the left) and a non-IP (on the right) approach.

link-layer BLE connection needs to be established, with the 6LBR having to know in advance whether the BLE peripheral at hand supports IPv6 over BLE. The Bluetooth SIG defines the Internet Protocol Support Profile (IPSP) to achieve this [7]. The BLE peripheral implements the Internet Protocol Support Service (IPSS), which defines a specific UUID that can be used as AD type in an BLE advertisement. By checking this AD type field, the 6LBR can determine that a BLE peripheral supports IPv6 over BLE. Once connected, an L2CAP channel is established on top, to offer fragmentation capabilities between BLE and a higher layer protocol. Finally, the higher layer protocols are set up above. Existing methods described in other 6LowPAN standards [8] [9] are leveraged to limit the overhead of stateless autoconfiguration, neighbor discovery and header compression.

III. Related work

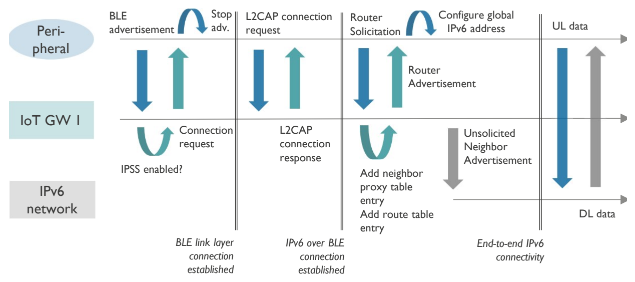
Two possible solutions for the IoT GW problem were presented in [3]: either provide end-to-end IP connectivity or configure the IoT GW to act as a proxy for the BLE profile, on a higher layer. The research done in [10] describes how end-to-end IP connectivity can be established, using CoAP and MQTT on top. It only describes a static network configuration, not tackling the dynamic networking environment incited by mobile peripherals within a network of IoT GW. The proxy solution has also been thoroughly researched, by employing the Network Service Discovery (NSD) capabilities of Android smartphones in order to configure a smartphone as a bridge between the BLE peripheral and the Internet [11]. However, such a proxy/bridge approach is not application-agnostic and there is no mention of handover capabilities.

A different approach described in [12] attempts to set up a remote BLE connection across an IP network. Here a new layer is added to the BLE stack, allowing a node (client) in an IP network to leverage on another node (proxy) in that network to set up a BLE connection to a nearby BLE peripheral. The client node is able to communicate with the BLE peripheral and maintains the BLE connection through the proxy node. The downside here is that much more control traffic is to be expected on the IP network and the implementation effort to configure a new layer in the BLE stack could prove to be too cumbersome compared to the alternative non-IP approach presented in this paper. There is also no mention of handover capabilities.

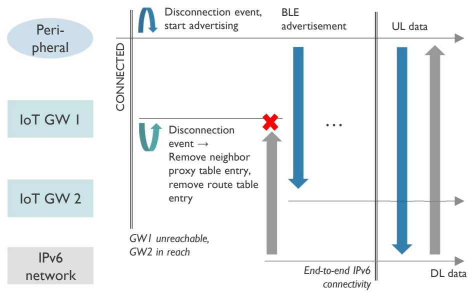
Today, most research on handover capabilities in low-power IoT networks has been limited to other wireless technologies than BLE. For example, the work in [13] describes how this can be achieved for IEEE 802.15.4e TSCH networks. Existing research on BLE handover provides insight on how to transfer the state of the connection during the handover process between gateways [14]. To the best of our knowledge, no research has been done before on handover capabilities within a network of BLE-enabled IoT GWs, that considers application-agnostic connectivity with an existing IP network by means of two separate architectures.

IV. A BLE-based multi-gateway network infrastructure

For the design of our BLE-based multi-GW network infrastructure, we consider two variants, both able to overcome the described GW problem. These two approaches are shown in Figure 2. The design on the left employs the IPv6 over BLE specification [6] to achieve end-to-end IP connectivity between the BLE peripheral and an application server in an existing IPv6 network, with the GW acting as an IPv6 router in between. The design on the right employs custom forwarding on the GW, in order to maintain native BLE communication between the peripheral and the GW while still offering a BLE-enabled handover infrastructure. It requires a dedicated controller within the existing IPv6 network, capable of communicating with each IoT gateway and maintaining a dynamic mapping between BLE peripherals and their current


Peri-pheral
IoT GW 1
IPv6 network
BLE advertisement
Stop adv.
Connection request
IPSS enabled?
L2CAP connection request
L2CAP connection response
Router Solicitation
Configure global IPv6 address
Router Advertisement
Add neighbor proxy table entry
Add route table entry
Unsolicited Neighbor Advertisement
UL data
DL data
BLE link layer connection established
IPv6 over BLE connection established
End-to-end IPv6 connectivity

(a) IPv6-based connectivity setup


Peri-pheral
IoT GW 1
IoT GW 2
IPv6 network
CONNECTED
Disconnection event, start advertising
BLE advertisement
Disconnection event →
Remove neighbor proxy table entry, remove route table entry
...
GW1 unreachable, GW2 in reach
End-to-end IPv6 connectivity
UL data
DL data

(b) Handover

Fig. 3: Connection setup and handover in the IPv6-based multi-gateway network infrastructure.

corresponding GW. Both approaches, further called IP and non-IP, should be able to achieve seamless handover within the network of IoT gateways. The following subsections describe in more detail the operation of both approaches.

A. IP based architecture

BLE peripheral. The BLE peripheral needs to mold its native BLE stack in order to support IPv6 over BLE. This requires more resources from the already resource-constrained peripheral. The peripheral generates a link-local address derived from the device's BLE MAC address and a global IPv6 address based on a common prefix. In order to move around inside a network of IoT GWs and perform handovers, the peripheral should always be in one of two states: a) in connection with an IoT GW, or b) advertising its presence to nearby IoT GWs using BLE advertisements.

IoT GW. The IoT GW should also be able to support the IPv6 over BLE stack, while at the same time taking on the role of IPv6 router and neighbor proxy towards the existing IPv6 network. The role of router implies that the GW forwards IPv6 packets from the BLE peripheral to the IPv6 network and vice versa. As proxy neighbor, the GW should answer Neighbor Solicitation (NS) packets for the BLE peripheral directly, informing the sender that the peripheral is reachable via this GW. This is done by answering with a Neighbor Advertisement (NA) packet.

To ensure that the peripheral acquires a global IPv6 address usable within the aforementioned IPv6 network, *stateless address autoconfiguration* is used [8]. In practice, this means that the peripheral sends a Router Solicitation (RS) packet towards the IoT GW, after which the GW answers with a Router Advertisement (RA) packet. This RA packet contains the prefix of the IPv6 network. Upon receiving this RA packet, the peripheral generates the global IPv6 address based upon his link-local address and the received prefix.

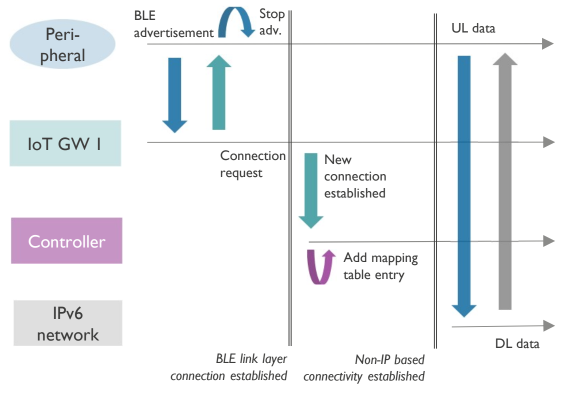
The peripheral should forward all its packets directly to the IoT GW, without sending NS packets for the end device it wants to reach first. The achieve this, the IoT gateway should set the *on-link* flag in the RA packet to 0. Thereby, the peripheral has no known on-link prefixes [6]. Consequently, the peripheral forwards all its packets to its first-hop default router. This router is determined by the source address of the RA packet, hence the IoT GW's link-local address. The entire flow is depicted in Figure 3a. Due to the end-to-end IP connectivity, if the IPv6 network also has access to the Internet, access to the BLE peripheral can be extended from the application servers in the existing IP network to the Internet as a whole.

Handover. In case of handover, there will be a certain period of time the BLE peripheral becomes unreachable as illustrated in Figure 3b. When the peripheral has gone out of reach of the current GW, the connection will be lost, and downlink packets will no longer arrive. Any attempts of a node to verify reachability by NS packets will fail. Once the peripheral has roamed to another IoT GW, that GW will become the new neighbor proxy for the peripheral and answer NS packets with NA packets. However, the time between finishing the connection setup and the next attempt to enquire reachability via such an NS packet, can already be used to communicate between the peripheral and the node. Therefore, when a new connection is fully set up, the GW can already pro-actively broadcast an Unsolicited Neighbor Advertisement (U-NA) packet, informing the existing IPv6 network that the BLE peripheral is now reachable via this GW.

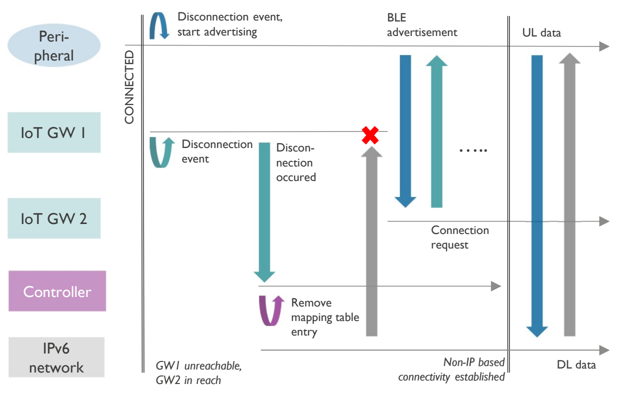
To be able to set up the connection with a new GW, each GW should continuously scan for BLE advertisements sent out by peripherals. Upon receiving such an advertisement, the GW should attempt to establish an IPv6 over BLE connection with the corresponding BLE peripheral. Whenever a connection occurs, the GW needs to configure the necessary route table entries and neighbor proxy table entry after which the U-NA packet is broadcasted into the existing IPv6 network. If a disconnection occurs, the corresponding route and proxy entries should be removed.

B. Non-IP based architecture

BLE peripheral. As no IP is used, a native BLE connection can be set up between the BLE peripheral and an IoT GW. Therefore, to set up IoT connectivity between the BLE peripheral and the controller, no extra implementation is needed on the peripheral apart from setting up native BLE communication. In order to achieve seamless handover, the peripheral is in one of two states at all times: a) in a BLE connection with an IoT GW, or b) advertising its presence



(a) Non-IP connectivity setup



(b) Handover

Fig. 4: Connection setup and handover in the non-IP multi-gateway network infrastructure.

using BLE advertisements.

IoT GW. The GW should expose the BLE peripheral towards the IP network in an application-agnostic way. Thus, the IoT GW only has a forwarding role and is unaware of the application layer connectivity between the BLE peripheral and the controller. This way, the GW can forward anything, without requiring any changes to its implementation whenever a new application layer protocol is used between the controller and the BLE peripheral.

Controller. The IoT GWs notify the controller when a new connection is established or a disconnection occurs. Therefore, the controller should keep a mapping table, linking each BLE peripheral to its current corresponding IoT GW. This is illustrated in Figure 4a. The peripherals are identified via an implementation-specific device ID. To expose the BLE peripheral to the Internet, the controller could act as a proxy or bridge, hereby translating the application layer to another protocol [3].

Handover. To achieve seamless handover for the BLE peripheral between several IoT GWs, each GW should continuously scan for BLE advertisements broadcasted by peripherals. Upon receiving such an advertisement, a GW should attempt to establish a native BLE connection with the corresponding BLE peripheral. Whenever a new connection is established or a disconnection occurs, it should notify the controller. This is shown in Figure 4b.

V. Handover approaches

To achieve handovers within a network of BLE-enabled IoT gateways, two approaches are considered, namely a passive and an active approach, both which are described in the remainder of this section.

A. *Passive handover*

This approach does not add any extra features to the proposed solutions. A BLE connection is terminated naturally after the *supervision timeout* has timed out. Only then, the BLE peripheral attempts to set up a new connection by advertising its presence using BLE advertisements. This approach is also used in Figure 3 and 4 to illustrate the impact of a handover on both architectures.

B. *Active handover*

When using the passive handover approach, a connection to a new and better GW will only be established when the previous connection was completely broken, i.e. when no traffic could be sent during the supervision timeout range. As such, it might happen that a peripheral remains connected even when the link quality becomes very poor. To avoid this, some proactive engagement is needed in order to guarantee a good BLE connection at all times. A naive approach could be to lower the supervision timeout dynamically. However, this is not very intelligent, as this could cause too much unnecessary handovers and short-lived connections because of too strict configurations.

In order to support an active handover approach, several changes have to be made to the BLE-enabled infrastructure, independent of it being IP or non-IP based, as both solutions utilize a BLE link-layer connection underneath. On top, now also the IP-based solution requires a dedicated controller. The active handover approach is illustrated in Figure 5 and the remainder of this subsection describes how this setup can be achieved.

The BLE peripheral maintains an advertiser role at all times, independent of its connection status. The IoT GWs use these periodic BLE advertisements to derive a Received Signal Strength Indicator (RSSI). These RSSI values give an insight on the peripheral's reachability with neighboring IoT GWs. All collected RSSI values are forwarded to the controller. The controller maintains a mapping table between the received RSSI values and the associated IoT GW and BLE peripheral. Concurrently, this mapping table is used to monitor the current state of the BLE connections and decides whether a specific BLE peripheral should handover to another IoT GW.

Such a handover decision is taken when two conditions are met: the connection to the current IoT GW is bad and a sufficiently good enough alternative IoT GW is available.

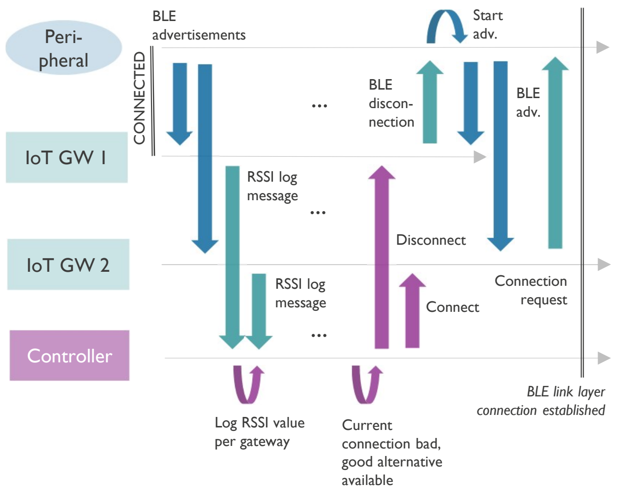


Fig. 5: A step by step overview on how the active handover process works.

The connection can be considered bad once the RSSI exceeds a certain RSSI threshold, meaning that the peripheral is not too far from the GW for the connection to be lost, but far enough to be of insufficient quality. Another GW can be considered a sufficient alternative, once its RSSI exceeds a certain threshold as well, indicating that the peripheral is actually close enough to that GW to establish a connection of sufficient quality. In order to achieve this, only the most recent RSSI values are stored and the decision is made based on a running average of these values. Figure 6 illustrates this handover process, which can be identified as a hysteresis phenomenon.

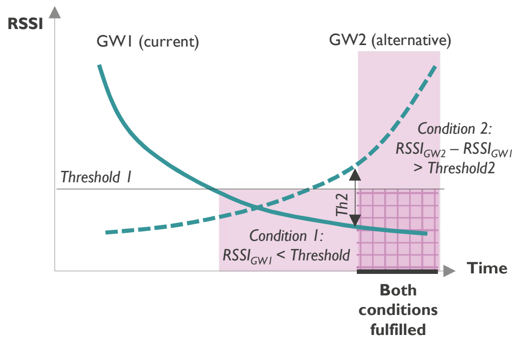


Fig. 6: An illustration of the hysteresis phenomenon incited by applying two separate RSSI thresholds to enable active handover.

When a handover decision is made, a *disconnect* message is sent by the controller to the old IoT GW and a *connect* message to the alternative IoT GW. The old IoT GW forwards this request to the BLE peripheral and the peripheral immediately terminates the connection (independent of the supervision timeout). Afterwards, the BLE advertisements can be answered with a connection request again, which is now done by the alternative gateway incited by the *connect* message.

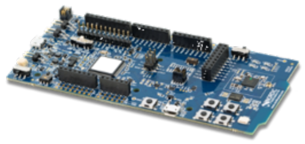
On top, when a BLE peripheral does not yet have a BLE connection with an IoT GW, the current best RSSI is can be used to find a suitable IoT GW and set up the connection. When an IoT GW naturally loses a BLE connection, which can occur if no suitable alternative was available, or a BLE connection failed to be set up, the controller is also notified. Such an event indicates that the BLE peripheral has moved out of range of the BLE-enabled infrastructure or the infrastructure contains a blind spot.

VI. Implementation details

The hardware used in the implementation is shown in Figure 7a and b. In Figure 7a, a nRF52840 Development Kit from Nordic Semiconductor is shown. Combined with Nordic's proprietary Software Development Kit (SDK) and SoftDevice, this device is capable of implementing a complete BLE v5.0 supported stack as well as the IPv6 over BLE stack adjustment. In Figure 7b, a Raspberry PI 3 is shown. A Raspberry PI 3 has a dedicated Linux based OS (Raspbian) available and implements several network interfaces, including Bluetooth. To communicate with this Bluetooth interface, an offical Linux Bluetooth protocol stack (BLueZ) is available. BLueZ offers several Bash commands to easily interact with the Bluetooth stack underneath. However, regardless of the Bluetooth hardware used underneath, the BLueZ stack does not yet offer support for BLE v5.0 (limited to BLE v4.2).

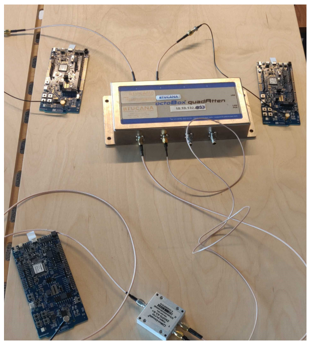
Section VII evaluates both handover approaches. To emulate a mobile BLE peripheral performing a handover from one gateway to another gateway, in a controller manner, a manual attenuator is used. The experimental setup is depicted in Figure 7c . Two boards act as IoT gateway and a third board acts as mobile BLE peripheral. The RF signals of the gateways and the BLE peripheral are routed directly to the attenuator, using SWF connectors.

The implementation follows the guidelines described in Section IV. The remainder of this section elaborates on the implementation specifics for each aspect of the IP and non-IP architecture.



(a) nRF52840 Development Kit from Nordic Semiconductor

(b) Raspberry PI 3



(c) Setup with controllable attenuator for handover emulation

Fig. 7: The hardware used in the implementation of both solutions and the setup for conducting handover experiments.

A. *IP based solution*

To be able to answer RS packets coming from a BLE peripheral, the *radvd* daemon is available. This daemon is used to answer incoming RS packets from the BLE peripheral, with RA packets containing a pre-configured prefix and the *on-link* flag set to 0. Next, the BLE peripheral should be able to roam from one IoT GW to another. To achieve this, each IoT GW runs an instance of the *bluetooth_6lowpand* daemon. This daemon uses the BLueZ C API underneath and is capable of commissioning IPv6 over BLE connections automatically. Currently, two options are available to achieve this automated connectivity: Authentication using WiFi keys or whitelisting specific BLE peripherals via their MAC address. This implementation uses the latter option. Combined with a custom script that sets up all other configuration necessary when a new connection is established or when a disconnection occurs, a seamless passive handover implementation is achieved. The BLE peripheral's role is enacted using a nRF52840 board and the other roles are enacted using Raspberry PI's.

B. *Non-IP based solution*

The BLE peripheral implements a custom BLE service, capable of providing bidirectional non-IP based BLE communication between a BLE peripheral and an IoT GW. The IoT GWs and the controller set up a bidirectional, socket-based communication link with each other. The GWs set up BLE connections with BLE peripherals in range. The controller identifies these BLE peripherals using a custom DEV ID, based on their MAC address. The BLE peripheral's role is enacted using a nRF52840 board and the controller's role is enacted using a Raspberry PI. The IoT GW combines a nRF52840 board with a Raspberry PI, in order to support a sophisticated custom BLE implementation while still offering an IP interface towards the controller.

VII. Evaluation

This section aims to evaluate the active and passive handover process, highlighting the main BLE parameters which impact its performance. Both handover approaches are evaluated using the experimental setup depicted in Figure 7c. The results of those experiments are shown in Figure 8 and indicate that the performance of the handover process is predominantly impacted by two steps: disconnecting from the current GW and setting up a connection with the alternative GW. The remainder of this section further describes these two steps for both approaches.

A. *Passive handover*

Disconnecting from the current GW. For passive handover, this part consists of waiting until the supervision timeout has timed out and has been indicated as **A** in Figure 8a and 8c. Both BLE connection entities maintain such a timeout and reset it whenever a BLE packet is received. According to the BLE standard, the value of this timeout has to be a multiple of 10 ms, between 100 ms and 32.0 s. It should be larger than *(1 + slave latency) * connection interval * 2*. Assuming no slave latency is used, the minimal supervision timeout is still twice the connection interval. The size of the connection interval depends on the application throughput demands and the energy constraints of the BLE peripheral. The supervision timeout should not be too low, to avoid unnecessary triggering of the handover procedure. However, a too high supervision timeout can have a significant impact on the handover latency, as shown by Figure 8e.

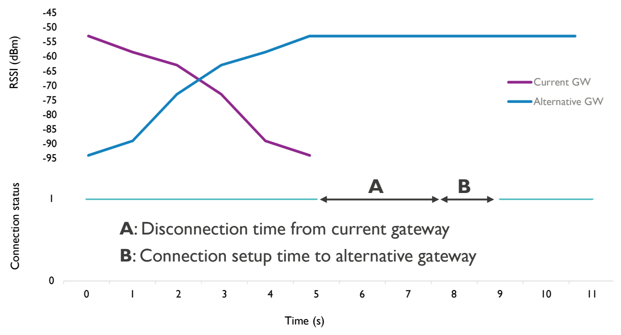
Setting up a connection with the alternative GW. Once the BLE peripheral's supervision timeout has timed out, it transitions back to the *advertising* state, broadcasting connectable advertisements at a certain advertising interval (indicated as **B** in Figure 8a and 8c). This interval consists of a manually configurable interval and a pseudo-random delay (0 - 10 ms). The configurable interval should be a multiple of 0.625 ms, between 20 ms and 10,485.759375 s. A GW that is not in a connection or still has room for another BLE connection, is in the *initiating* state and attempts to set up a connection upon receiving the connectable advertisements from the BLE peripheral, as shown in Figure 8c. The GW can use a predefined whitelist of MAC addresses to know whether it should attempt to connect to the BLE peripheral whenever such an advertisement is received. The GW shifts between the advertisement channels, actively scanning during the scan window. The scan interval should be less than 40.96 s and the scan window should be less than or equal to the scan interval.

Several BLE parameter combinations are evaluated in Figure 8e. The advertising interval should be smaller or equal to the scan window, because otherwise it cannot be guaranteed that the GW will be able to receive a connectable advertisement in an acceptable time window [4]. A higher scan duty cycle (*scan window/scan interval*) has a positive impact on the handover latency but a negative impact on the energy consumption of the GW. A duty cycle of 100 % means that the scanner is continuously scanning. The gateway will generally be mains-powered, so the negative impact of this on energy consumption is negligible. Further, utilizing continuous scanning minimizes the energy consumption of the BLE peripheral as its advertisements are received much faster [5].

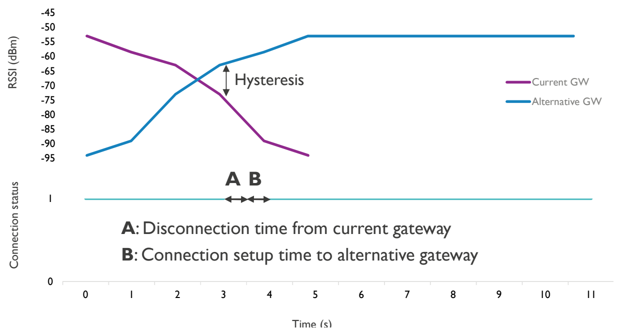
If continuous scanning cannot be used (due to energy limitations of the GW), the scan window should at least be high enough as opposed to the scan and advertising interval. Otherwise, a higher handover latency can be expected and the energy consumption of the BLE peripheral goes up as well, due to longer periods during which advertising has to take place [5]. Lastly, a longer advertising interval can limit the energy consumption of the BLE peripheral, but can also cause a higher handover latency.

B. *Active handover*

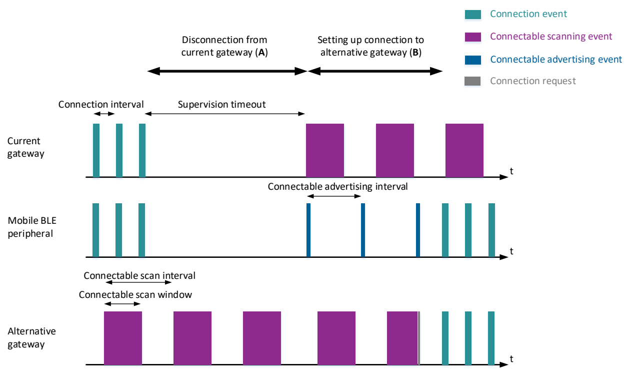
Disconnecting from the current GW. As explained, the current GW will receive a disconnection message from the controller once a roaming decision has made. Figure 8f indicates the RSSI threshold upon which a BLE connection can be considered to be of insufficient quality. After receiving



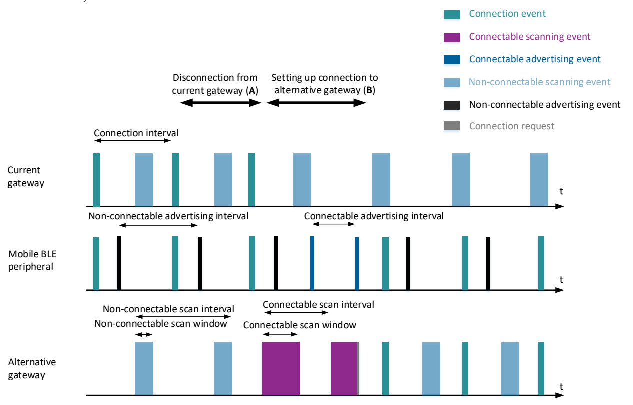
(a) Experimental passive handover measurement using 2 BLE GWs, 1 peripheral and an attenuator for controlling the GWs signal strengths. Supervision timeout = 4000ms, advertising interval = 80ms, scan interval = 500ms, scan window = 300ms



(b) Experimental active handover measurement using 2 BLE GWs, 1 peripheral and an attenuator for controlling the GWs signal strengths. Supervision timeout = 4000ms, advertising interval = 80ms, scan interval = 500ms, scan window = 300ms


Connection event
Connectable scanning event
Connectable advertising event
Connection request
Disconnection from current gateway (A)
Setting up connection to alternative gateway (B)
Connection interval
Supervision timeout
Current gateway
Connectable advertising interval
Mobile BLE peripheral
Connectable scan interval
Connectable scan window
Alternative gateway
t

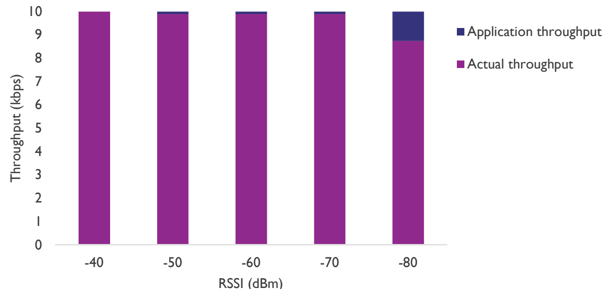
(c) Impact of BLE parameter settings on the passive handover performance


Connection event
Connectable scanning event
Connectable advertising event
Non-connectable scanning event
Non-connectable advertising event
Connection request
Disconnection from current gateway (A)
Setting up connection to alternative gateway (B)
Connection interval
Current gateway
Non-connectable advertising interval
Connectable advertising interval
Mobile BLE peripheral
Non-connectable scan interval
Non-connectable scan window
Connectable scan interval
Connectable scan window
Alternative gateway
t

(d) Impact of BLE parameter settings on the active handover performance

Superv. timeout	Conn. adv. interval	Conn. scan interval	Conn. scan window	Disconnection latency (**A**)	Connection setup latency (**B**)
4000	80	500	300	4000	56
4000	80	500	500	4000	1
4000	80	500	90	4000	201
1000	200	500	300	1000	80
500	800	500	300	500	321

(e) Theoretical assessment of the expected passive handover performance for given BLE parameters (all values are expressed in ms) [4]



(f) Throughput of the BLE connection based on RSSI measurements

Fig. 8: Passive and active handover performance assessment and analysis.

this message, the GW immediately attempts to send a disconnection request to the BLE peripheral. Upon receiving this request, the peripheral answers with a disconnection acknowledgment. Thereby, the disconnection process is completed. At first glance, this process is much faster than it is with passive handover, because the disconnection process is independent from the supervision timeout. However, other BLE parameters can influence how fast the disconnection occurs.

A shorter connection interval can lower the latency of sending a disconnection request and receiving the acknowledgment. It should be a multiple of 1.25 ms, between 7.5 ms and 4 s. However, both entities are not only in a *connection* state. The GW combines the *connection* state with the *scanning* state, listening for incoming non-connectable advertisements in order to send new RSSI log messages to the controller. The BLE peripheral combines the *connection* state with the *advertising* state, advertising non-connectable advertisements to nearby GWs (including the current GW), which these GWs use to send RSSI log messages to the controller. How these combinations are handled, is implementation-specific to the BLE stack. However, it is clear that a specific combination of the connection interval, non-connectable scan interval/window and non-connectable advertisement interval, can either positively or negatively impact the throughput of the connection (i.e. how fast a disconnection request/acknowledgment is send), as well as the rate at which new RSSI log messages are send to the controller. For the settings used in Figure 8b, the disconnection time was observed to be significantly lower than for the passive handover case with similar settings, as shown in Figure 8a.

Setting up a connection with the alternative GW. Once the disconnection has been finalized, the BLE peripheral switches from a *connection/advertising* state to solely an *advertising* state, broadcasting connectable advertisements. In the meantime, the alternative GW has received a connect message from the controller and has switched from non-connectable to connectable scanning, specifically for the MAC address of the BLE peripheral at hand. From this point, the handover process is similar to passive handover.

Potential optimizations. Currently, the BLE peripheral constantly needs to advertise. This increases energy consumption and decreases the quality of the BLE connection as well. A possible improvement could employ a proactive decision on when to start and stop advertising. The peripheral can monitor the quality of the current connection (i.e. via RSSI) and decide via these indicators whether it is necessary to advertise its presence or not.

During a handover process, the BLE peripheral is not reachable from application servers. This implies possible data loss. The newest BLE standard allows a BLE peripheral to be in a connection with multiple BLE centrals at once. This could be employed within the active handover approach, by allowing the current GW to keep its connection with the BLE peripheral, until the new connection with the alternative GW is established. That way, the communication on top always has a BLE connection it can utilize so no data loss occurs.

Finally, to avoid constant switching between connectable and non-connectable scanning, two separate BLE modules could be used on the same GW. One module is in charge of connectable scanning and maintains existing BLE connections. The other module performs only non-connectable scanning, continuously sending RSSI log messages to the controller.

VIII. Conclusion and future work

In this paper, two architectures are proposed to realize a BLE-based multi-GW network infrastructure that is able to provide bidirectional connectivity to mobile BLE peripherals. The IP based architecture offers seamless end-to-end IP connectivity between a mobile BLE peripheral and an existing IP network. It limits custom implementation needs, but requires more resources from the already constrained mobile BLE peripheral. The non-IP based architecture requires less resources from the BLE peripheral as it employs native BLE communication towards the IoT gateway. However, custom implementation needs are much higher, as a dedicated controller is introduced within the existing network that needs to maintain the state of the infrastructure and the currently associated mobile BLE devices.

Next to this, two approaches are considered to perform handover within these solutions. The passive handover approach does not need any extra implementation, but the handover latency can become quite high and a bad BLE connection is still maintained even if a better alternative GW is available nearby. Active handover offers a solution by proactively terminating a bad connection when a better alternative GW is available. As such, it can limit the handover latency. As a downside, this approach requires more extensive custom implementation, implies a higher energy consumption and can limit the available throughput.

Future work could look into the impact of multiple BLE peripherals within the infrastructure. Further, a study can be done on security implications within the solution. Finally, a comparison could be done between the connection-based handover solutions proposed in this paper and advertising-based handover solutions such as Bluetooth Mesh networks [15].

Acknowledgment

This research was funded by the ICON project MAGICIaN. MAGICIaN is realized in collaboration with imec, with project support from VLAIO and Innoviris. Project partners are imec, Orange Belgium, Televic Healthcare, Re- store and Citymesh.

References

[1] A. Nikoukar, S. Raza, A. Poole, M. Gne and B. Dezfouli, "Low-Power Wireless for the Internet of Things: Standards and Applications", IEEE Access, vol. 6, 2018.

[2] Bluetooth Core Specification: 5.1, Bluetooth Special Interest Group, Kirkland, Washington, U.S., 2019.

[3] T. Zachariah, N. Klugman, B. Campbell, J. Adkins, N. Jackson, and P. Dutta, "The Internet of Things Has a Gateway Problem", In Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications (HotMobile '15) ACM, New York, NY, USA, 2015.

[4] J. Liu, C. Chen and Y. Ma, "Modeling Neighbor Discovery in Bluetooth Low Energy Networks", in IEEE Communications Letters, vol. 16, 2012.

[5] J. Liu, C. Chen, Y. Ma and Y. Xu, "Energy Analysis of Device Discovery for Bluetooth Low Energy", 2013 IEEE 78th Vehicular Technology Conference (VTC Fall), Las Vegas, NV, 2013.

[6] J. Nieminen, T. Savolaienen, M. Isomaki, B. Patil, Z. Shelby,and C. Gomez, "IPv6 over BLUETOOTH(R) Low Energy", 2015.

[7] Bluetooth Special Interest Group, "Bluetooth Internet Protocol Support Profile Specification Version 1.0.0", 2014.

[8] Z. Shelby, S. Chakrabarti, E. Nordmark and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", 2012.

[9] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", 2012.

[10] C. Lin, K. Liao and C. Chang, "An Experimental System for MQTT/CoAP-based IoT Applications in IPv6 over Bluetooth Low Energy". Journal of Universal Computer Science, vol. 24, 2018.

[11] J. Rossey, I. Moerman, P. Demeester, J. Hoebeke, "Wi-FI helping out Bluetooth Smart for an improved home automation user experience", IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT), 2016.

[12] N. Rozhnova, M. Buob and R. Douville, "Application-agnostic remote access for Bluetooth Low Energy," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Honolulu, HI, 2018.

[13] J. Haxhibeqiri, A. Karaagac, I. Moerman, and J. Hoebeke, "Seamless handover and guaranteed communication using a synchronized single-hop multi-gateway 802.15.4e TSCH network", Ad Hoc Networks, 2018.

[14] S. R. Hussain, S. Mehnaz, S. Nirjon, and E. Bertino. "SeamBlue: Seamless Bluetooth Low Energy Connection Migration for Unmodified IoT Devices", In Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks (EWSN 17), Junction Publishing, USA, 2017.

[15] Mesh Profile Specification: 1.0, Bluetooth Special Interest Group, Kirkland, Washington, U.S., 2017.