Transmitting IPv6 packets over Bluetooth low energy based on BlueZ

Conference Paper · January 2013

Haolin Wang
Chongqing Medical University

Minjun Xi
Beihang University (BUAA)

Jia Liu
Zhigu Tech.

Canfeng Chen
Nokia Research Center (NRC)

Transmitting IPv6 Packets over Bluetooth Low Energy based on BlueZ

Haolin Wang*, Minjun Xi**, Jia Liu**, Canfeng Chen**

*School of Software Engineering, University of Science and Technology of China

**Nokia Research Center

hl.wang@outlook.com, minjun.xi@nokia.com, ext-jia.17.liu@nokia.com, canfeng-david.chen@nokia.com

***Abstract*— Recent years have witnessed the development of wireless sensor networks and Internet of Things (IoT) technologies, and IPv6-based solution for sensor networks has attracted more and more attention which can enable end-to-end IP communication via the Internet. As far as we know, we implemented the first prototype system to transmit IPv6 packets over Bluetooth low energy (BLE) based on BlueZ, the official Bluetooth stack of Linux. To adapt IPv6 packets to BLE link, in addition to the implementation of segmentation and reassembly (SAR), header compression (HC) is effective to improve the transmission efficiency and achieve lower power consumption. The 6LoWPAN workgroup of Internet Engineering Task Force (IETF) has proposed several standards and drafts to specify the header compression scheme for IPv6 packet delivery in low power wireless networks. Based on the compression format defined in RFC6282, we proposed a new mechanism for the management of contexts, which can improve both power efficiency and time efficiency, extend the content of context information and provide better flexibility.**

***Keywords*— Bluetooth Low Energy, IPv6, Header Compression**

I. INTRODUCTION

Bluetooth low energy [1] is one of the latest technologies applied for IoT applications and it's a short-range wireless communication technology that aims at ultra-low power. It operates in the 2.4GHz ISM band and was adopted by Bluetooth core specification v4.0 in 2010. BLE is essentially a redesign of the Bluetooth standard which is suitable for sensors transmitting data infrequently or peripherals using asynchronous communication. To reduce power consumption, its physical layer is redesigned and its state machine is optimized for its use cases. BLE defines 40 physical channels using GFSK modulation to achieve a data rate up to 1 Mbps and uses frequency hopping transceiver to reduce interference. The standby power consumption of BLE devices is ultra-low and the operating power consumption can be further reduced by the design of simplified protocols. Bluetooth is a connection-oriented protocol and BLE connection can be established very quickly. BLE supports star topology and a typical scenario for its use towards the Internet of Things is shown in Figure 1. In a BLE network, the sensor nodes can be a variety of smart devices and a smart phone or a mobile Internet device (MID) can be used as the router to provide Internet connectivity through its network interface.

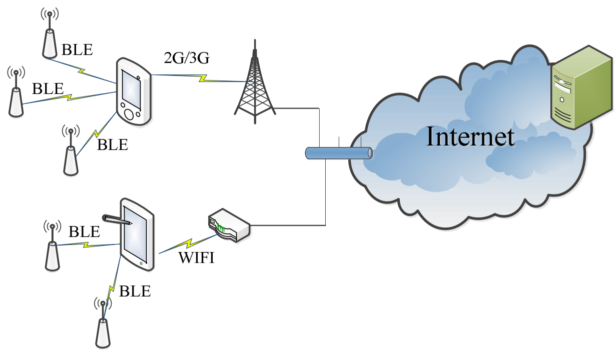

Figure 1. Star topology and a typical scenario of BLE

In fact, BLE is used in a wide range of medical, industrial, and consumer applications such as the heart rate sensors, security tags and proximity sensors. The number of BLE enabled smart devices is increasing rapidly.

IPv6 is the next version of Internet protocol, providing a huge address space for sensors. IPv6-enabled wireless sensor network (WSN) switches the sink node's role from proxy gateway to IPv6 router. Meanwhile, the use of IPv6 protocol in the routing layer can shield the difference of the link layer, so the same services can be provided based on IPv6 protocol over heterogeneous links.

A new IETF draft is already under way to specify the transmission of IPv6 packets over BLE [2] and the protocol stack model is shown in Figure 2. The logical link control and adaption protocol (L2CAP) provides protocol/channel multiplexing, per-channel flow control and error control. The 6LoWPAN adaption layer is defined for IPv6 over low-power wireless personal area networks (6LoWPAN) to handle encapsulated IPv6 packets. The functionality of header compression in this layer is being studied due to the fact that Bluetooth core specification v4.0 defines the maximal MTU as 23 bytes over the BLE user logical link, however a complete IPv6 header, if incorporated, takes 40 bytes, let alone the overhead of upper layer, such as 8 bytes UDP header. In typical WSN applications, the amount of sensor data is usually in the order of several bytes. To increase the percentage of valid data transmitted over the air-interface and improve the throughput of the application layer, it is necessary to implement header compression, which is particularly useful in such slow links.

UDP/TCP
IPv6
6LoWPAN adapted to BLE
BLE L2CAP
BLE Link Layer
BLE Physical

Figure 2. IPv6 over BLE Stack

To the best of our knowledge, we implemented the first lightweight IPv6 stack over Bluetooth low energy based on BlueZ, which is the official Linux Bluetooth protocol stack, including an IPv6/UDP header compression subsystem. Based on the existing IPv6 header compression standard for IEEE 802.15.4, we proposed a new mechanism to manage the contexts for BLE-based WSN.

The rest parts of the paper are organized as follows: the related protocols and IETF drafts are introduced in Section 2; the design and implementation of the lightweight IPv6 stack based on BlueZ is introduced in Section 3, including the overview, the solution for SAR, HC and the proposed context exchange mechanism; In section 4, we analysed the performance of the prototype system.

II. RELATED WORK

6LoWPAN is also an IETF working group which specifies the protocols to build IPv6-connected wireless sensor networks [3]. Although some 6LoWPAN core specifications are proposed for IEEE 802.15.4-based networks, most of the mechanism can be applied to BLE-based networks because these two technologies have many features in common. According to the IETF plan, the specification for transmission of IPv6 packets over BLE will become the new member of the 6LoWPAN family.

As we mentioned before, IPv6 header compression is almost the most important function of the adaption layer applied within such low power and lossy networks. IETF standard RFC6282 [4] proposes the compression scheme which relies on synchronized contexts between the communicating nodes. The fundamental of header compression is that the IPv6 header fields use some common values including some redundant information, which can be compressed or even fully elided. The compressed or elided information can be restored when expanding the packet. For example, the length field and the checksum field in IPv6/UDP header can be recalculated if the correctness of data can be guaranteed by the CRC check of link layer. And some uncertain fields can be replaced by short identifiers, for example, 128-bit IPv6 address can be replaced by 4-bit context identifier (CID) and restored with the help of shared contexts. RFC6282 specifies the compression format of IPv6 packets over IEEE 802.15.4-based networks. Both IPv6 header and UDP header can be compressed and the compressed format is shown in Figure 3.

0 8

0	1	1	TF	NH	HLIM	CID	SAC	SAM	M	DAC	DAM
SCI	DCI	1	1	1	1	0	C	P			
SRC PORT	DST PORT										

Figure 3. Compressed UDP/IP Header Format defined by RFC6282

The first 2-byte is the compressed IPv6 header using LOWPAN_IPHC encoding. These fields include Traffic class and Flow label (TF), Next Header (NH), Hop Limit (HLIM), Context Identifier Extension (CID), Source Address Compression (SAC), Source Address Mode (SAM), Multicast Compression (M), Destination Address Compression (DAC) and Destination Address Mode (DAM). The third byte is the context identifier extension used for stateful compression, consisting of source context identifier (SCI) and destination context identifier (DCI). In this way, the 4-bit context identifier (CID) can be used in the compressed IPv6 header instead of the 128-bit IPv6 address, or the IPv6 prefix. The next two bytes are the compressed UDP header using LOWPAN_NHC encoding, where these fields include Checksum (C), Port (P), and source port and destination port.

Another Internet draft [I-D.ietf-6lowpan-nd] [5] proposed the optimized neighbour discovery protocol for 6LoWPAN and provided the 6LoWPAN context option (6CO) as the only way for the router to distribute contexts in the WSNs as shown in Figure 4. However, it does not provide the solution for sensors to initialize the contexts or update the contexts. Actually, considering the characteristics of BLE, the neighbour discovery protocol can be further simplified for its simple star topology, besides, it's not necessary to bind the context management and neighbour discovery together. The MTU of BLE is much smaller than that of IEEE 802.15.4, so if the same context exchanging message is sent over BLE, the message has to be segmented and recombined. The direct result is that more BLE frames will be transmitted to deliver the single message, leading to more power consumption and longer latency. Though the data rate of BLE physical layer is 1Mbit/s, due to factors such as the key BLE parameters configuration, the data transmission in practical applications is much slower. Transmission of more BLE frames means higher latency to initialize or update the IPv6 connection over BLE.

Type	Length		
Context Length	Res	C	CID
Reserved			
Valid Lifetime			
Context Prefix			

Figure 4. 6LoWPAN Context Option format

III. DESIGN AND IMPLEMENTATION

BlueZ [6], the widely used Bluetooth stack, is chosen to implement the light weight IPv6 stack since its component is integrated with the Linux kernel and it provides support for core Bluetooth layers and protocols. The HC subsystem is also implemented with the context management mechanism proposed in this section.

A. The implementation based on BlueZ

As shown in Figure 1, a BLE network consists of sensor nodes and routers. Nokia N9, a Linux based smart phone integrated with TI WL1273 BLE chip is chosen as the router. And the keyfobs with TI CC2540 BLE chip are used as the sensor nodes. The lightweight IPv6 stack, uIPv6 [7] original integrated with Contiki, an open source OS for microcontrollers, has been ported to TI CC2540 keyfob to support IPv6. The Bluetooth core system contains two major subsystems, the Controller and the Host. In order to keep the interoperability between different Bluetooth subsystems, the common layer called "Host Controller Interface (HCI)" is defined. In Linux, most of the host part is implemented as kernel modules and the controller part is implemented in the chip provided by its vender.

To implement the lightweight IPv6 protocol stack avoiding the modification of existing IPv6 module of Linux kernel, an adaption layer is implemented below the IPv6 layer as the stack model shown in Figure 2. The components of BlueZ are shown in Figure 5 and the 6LoWPAN adaption layer is added to the BlueZ core as a relatively independent kernel module.

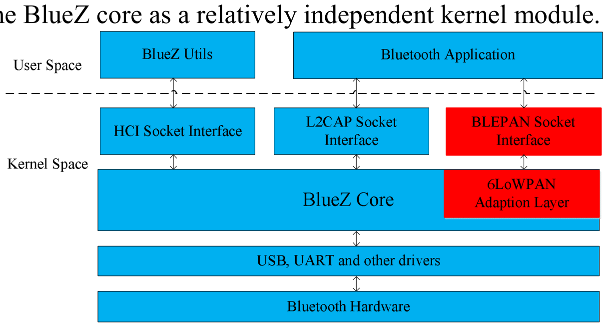

Figure 5. BlueZ Overview Diagram

In logical link control and adaption protocol (L2CAP) layer, the link type is set to BLE link and a new logical channel with channel ID 0x0007 is defined for transmitting IPv6 packets over BLE. Besides, to meet the requirements of application layer and IPv6 MTU, the SAR functionality is implemented in this layer by using L2CAP Information Frame (I-frame) defined in Bluetooth Core Specification. The 6LoWPAN adaption layer is used to handle the data flow between transport layer and L2CAP layer, and provide a network interface for upper layer applications at the same time. Both the IPv6 Header Compression and context management functionality is implemented in this adaption layer. The 6LoWPAN header compression system for BLE-based networks is shown in Figure 6.

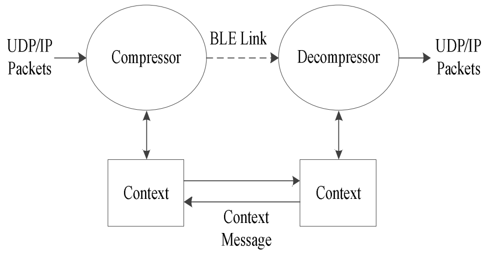

Figure 6. Header Compression System Diagram

B. Proposed Context Exchange Mechanism

One observed trend is that web service will be provided for the IoT through protocols based on REST concepts like the Constrained Application Protocol (CoAP) [8]. In WSNs, a node equipped with multiple sensors may bind with one or more IPv6 addresses that provide services. Working with web service provider, the vendor can put the IPv6 address for the predefined remote server into the sensors. Alternatively, one known server can provide the DNS-like services to distribute global IPv6 addresses for various web services. In each case, after the connection established between the sensor and the router, the sensor may need to register one or more remote server's IPv6 addresses that it will communicate with by sending the context information to the router. Afterwards, the router will answer with corresponding CIDs for received contexts as the response. After the context information is exchanged successfully, the sensor is able to communicate with the remote server through the IPv6 router. The procedure to update the context information is similar.

In addition to allowing the context exchange procedure initiated by the sensor, this mechanism can also expand the capability of compression. According to RFC6282, the fields which can't be compressed should be carried in-line following the compressed header. If not frequently changed, these fields can be stored in the context instead of being carried in-line. The extension of context information will be helpful to optimize the header compression for some specific use cases. For example, long strings will be used as the payload if a configuration file formatted in JSON or XML is transmitted over BLE [9], which can also be stored in the context to compress the content of payload. The content type field is used for this purpose as we described in the following paragraphs.

Since the proposed mechanism is defined for BLE-based WSN and implemented in the 6LoWPAN adaption layer, it will not affect the interoperability of the existing IPv6 infrastructures. Internet Control Message Protocol version 6 (ICMPv6) is an integrated part of IPv6 for transmitting control packets. Based on the general format of ICMPv6 message, the message for exchanging context is defined as Figure 7.

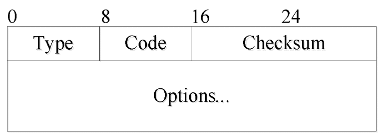

Figure 7. Context Exchange Message

The type field could be set to 100, 101, 200 or 201, which are reserved for private experimentation. The code field is set to 0 as the default value. All the context information is maintained by the router and hence this procedure must be a round-trip transaction in most cases. The context notification option (CNO) and context acknowledgment option (CAO) are specified as Figure 8 and Figure 9.

0	8		16				24
Type	CT	Length	M	R	C	LCID	Lifetime
Context							

Figure 8. Context Notification Option

Type: 4-bit option type.

CT: 4-bit Context Type identifier used to indicate that the content of the context is IPv6 addresses or other types of fields that should be carried in-line according to RFC6282. The function of this field can also be extended to indicate the fields of some application layer protocols, or even the content of the common payload according to specific application scenarios.

Length: 8-bit unsigned integer, the length of context field in unit of bit.

M: 2-bit method flag, to add, update or delete a context.

R: 1-bit flag used to indicate that its role is a router or a sensor.

C: 1-bit flag used to indicate if the context can be used for compression.

LCID: The local identifier for the following context from the viewpoint of the sender.

Valid lifetime: 1 byte, valid lifetime of the context information.

Context: The context field can be an IPv6 prefix, a full address or some other fields. All the fields appear in the same order as a normal IPv6 header.

Dual context identifier is designed to minimize the message size transmitted for the interaction between devices. The acknowledgement option is optional if the corresponding CNO is sent by the router since this CNO's LCID is allocated for public use.

0		8	
Type	Status	LCID	GCID

Figure 9. Context Acknowledgement Option

Type: 4-bit option type.

Status: 4-bit status field indicates the completion of an operation or returns the error code.

LCID: Local context identifier received from the CNO, it is used for the receiver to find corresponding local context.

GCID: Global context identifier allocated by the router is used for compression of IPv6 header. The router maintains the context table for all the sensors connected to it.

In summary, the proposed mechanism provides a solution for context exchange initiated by either the router or the sensor, which does not rely on neighbour discovery and allows the extension of context content. We defined different methods to add, update, and delete the context which provides better flexibility to manage the contexts. A sensor initiated context exchange procedure is shown in Figure 10. When the sensor is establishing the IPv6 connection to the router, the proposed context exchange message (CEM) with one or more CNOs will be sent to the router. Each CNO consists of a context and its LCID. Immediately, the router responds with CEM containing corresponding CAOs. Each option consists of a LCID and a GCID. Once the sensor receives the acknowledgement, it will save the GCID for every context. After that, the GCID can be used for compression and decompression.

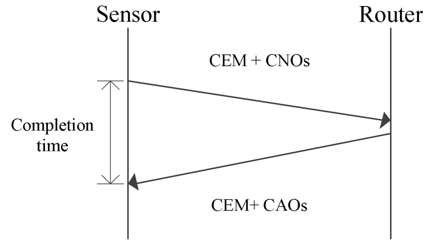

Figure 10. Sensor Initiated Context Exchange

The router initiated context exchange procedure is shown in Figure 11. The acknowledgement message is optional as we mentioned before.

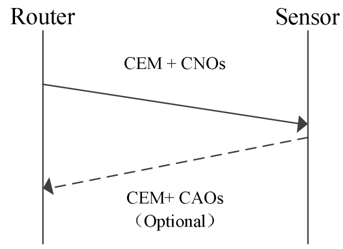

Figure 11. Router Initiated Context Exchange

IV. PERFORMANCE ANALYSIS

With the current implementation, the function integrity is checked by setting up the experiment scenario as shown in Figure 1. Nokia N9 connects to an IPv6 server through a WIFI router while N9 connects to a CC2540 keyfob on a BLE link. The function of the IPv6 stack is verified by sending and receiving IPv6 packets over BLE network interface with UDP servers and clients.

The following evaluation aims at the function of the 6LoWPAN adaption layer. The performance of header compression is tested by sending requests from the server to the keyfob, and the keyfob will respond with messages containing sensor data (e.g. the temperature). Every compressed packet is so small that it can be transmitted in a single BLE frame. The header length of IPv6 and UDP header is 48 bytes and the compressed IPv6 and UDP header is 5 bytes (Table 1).

TABLE 1. IPv6 AND UDP HEADER LENGTH

Origin length	Compressed length
48 bytes	5 bytes

The time consumption of header compression is calculated by checking the Linux kernel log and the experiment result is shown in Figure 12. The measurement is influenced by the outputting of debugging information for development, the time accuracy and task scheduling. The average time to compress the header is 0.086 milliseconds and the average time to decompress the header is 0.311 milliseconds. The decompression process consumes more time because that it has to allocate memory and calculate the omitted fields to reconstruct the UDP/IP packets, which also results in its instability.

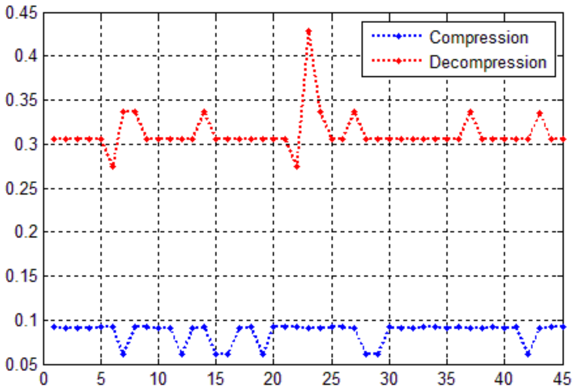

Figure 12. Time consumed for header compression on router

Especially for the comparison of sensor initiated context exchange procedure, two different schemes can be used to exchange three full IPv6 addresses as examples.

Scheme 1: We suppose that 6CO can also be carried in the Router Solicitation (RS) message. The sensor sends an RS with three 6COs, then the router replies with the Router Advertisement (RA) message containing three 6COs.

Scheme 2: The sensor sends a CEM with three CNOs, and then the router replies with a CEM containing three CAOs.

The message size and corresponding number of segmented BLE frames are listed in Figure 13.

Scheme	Content	Message size (Byte)	Number of I-frames
Scheme 1	RS+3COs	92	5
	RA+3COs	84	5
Scheme 2	CEM+3CNOs	68	4
	CEM+3CAOs	14	1

Figure 13. The comparison of message size for two schemes

From Figure 13, we can see that Scheme 2 outperforms Scheme 1, because more transmission frames will result in higher latency and more power consumption. As defined in the Bluetooth core specification, the link layer only transmits data in connection events through data channels. If a single message divided into several frames can't be transmitted in a single connection event, the rest frames should be transmitted in the next connection event. So the latency may be much higher with some specific parameters configuration, which is also depends on the lower layer limitation. For example, the TI stack limits the maximum number of frames to be transmitted during a 10ms connection event to 4 [10]. Under such circumstance, to finish transmitting 5 frames in scheme 1 consumes another 10ms relative to transmitting 4 frames in scheme 2. In addition, BLE uses time division multiplexing (TDM) between different physical channels to support concurrent operations so the latency will be much higher if the router connected with multiple devices.

Since the CC2540 keyfob is powered by a coin cell battery, its power consumption is always a concern. The BLE power consumption modelling for CC2540 is described in [10]. The power consumption during a connection event consist of energy to wake up, transmit (E_{tx}), receive (E_{rx}), wait for required number of IFS (E_{ifs}) and do post-processing. The power consumption to wake up and do post-processing is roughly the same for similar computational complexity of two schemes so that we compare the energy consumed for transmitting and receiving which phases lead to most power consumption for context exchange.

$$E_{tran} = E_{tx} + E_{rx} + E_{ifs}$$
$$= P_{tx}D_{tx}l_{tx} + P_{rx}D_{rx}nl_{hdr} + P_{ifs}D_{ifs}(2n-1) \quad (1)$$
$$E_{recv} = E_{rx} + E_{tx} + E_{ifs}$$
$$= P_{rx}D_{rx}l_{rx} + P_{tx}D_{tx}nl_{hdr} + P_{ifs}D_{ifs}(2n-1) \quad (2)$$

In the equations above, n, l_{hdr}, l_{tx}, l_{rx} denotes the number of I-frames, the length of BLE link layer header in unit of bit, the number of bits to be transmitted and the number of bits to be received in a connection event. The constants are shown in Table 2 and the variables n, l_{tx}, l_{rx} can be calculated according to Figure 13.

TABLE 2. POWER DRAW OF SELECTED PHRASES

Phase	Power Draw	Duration
RX	P_{rx}=66mW	D_{rx}=8us/bit
IFS	P_{ifs}=45mW	D_{ifs}=150us
TX	P_{tx}=84mW	D_{tx}=8us/bit

The power consumption of two schemes to transmit and receive packets can be calculated as shown in Table 3. From the perspective of CC2540 keyfob, scheme 2 conserves about half of the energy to transmit and receive frames for an interaction to exchange contexts.

TABLE 3. POWER CONSUMPTION OF TWO SCHEMES

Scheme	Energy (mJ)
Scheme 1	1.662
Scheme 2	0.844

V. CONCLUSIONS

The lightweight IPv6 stack along with the proposed context exchange mechanism for IPv6 capable BLE sensor networks has been implemented and evaluated. The 6LoWPAN-optimised IPv6 header compression and decompression over BLE-based networks is enabled for the end-to-end IPv6 communication. The proposed mechanism completes the exchange of context information with better flexibility and reduces the frames to be transmitted, the latency and the power consumption. The experiment results proved that the header compression and decompression consume very little time while they can reduce the number of link layer packets and improve the transmission efficiency.

REFERENCES

[1] Bluetooth SIG. Bluetooth specification version 4.0. June 2010.

[2] J. Nieminen, Ed., T. Savolainen, M. Isomaki, Z. Shelby and C. Gomez. Draft-ietf-6lowpan-btle-06 Transmission of IPv6 packets over bluetooth low energy. March 2012.

[3] C. Bormann. Draft-bormann-6lowpan-roadmap-01 6LoWPAN Roadmap and Implementation Guide. March 26, 2012.

[4] J.Hui, Ed. and P. Thubert. RFC6282 Compression format for IPv6 datagrams over IEEE 802.15.4-based networks. September 2011.

[5] Z. Shelby, Ed. S. Chakrabarti and E. Nordmark. Draft-ietf-6lowpan-nd-18 Neighbor discovery optimization for low power and lossy networks. October 2011.

[6] J. Beutel, M. Krasnyanskiy, 2001, Linux BlueZ Howto.

[7] Mathilde Dirvy, Julien Abeille, Patrick Wetterwald, et al.. Poster abstract: making sensor networks IPv6 ready. Proceedings of the 6th ACM conference on Embedded network sensor systems. 2008.

[8] A. Castellani, M. Gheda, N. Bui, M. Rossi, and M. Zorzi. Web Services for the Internet of Things through CoAP and EXI. In Proc. ICC, Kyoto, Japan, 2011.

[9] J. Nieminen, Ed., B. Patil, T. Savolainen, et al.. Constrained Application Autoconfiguration. Draft-nieminen-core-service-discovery-02. March 12, 2012.

[10] Matti Siekkinen, Markus Hiienkari, Jukka K. Nurminen and Johanna Nieminen. How Low Energy is Bluetooth Low Energy? Comparative Measurements with ZigBee/802.15.4. WCNC 2012 Workshop on Internet of Things Enabling Technologies, Embracing Machine-to-Machine Communications and Beyond. 2012.