Wi-LE: Can WiFi Replace Bluetooth?

Ali Abedi
University of Waterloo
ali.abedi@uwaterloo.ca

Omid Abari
University of Waterloo
omid.abari@uwaterloo.ca

Tim Brecht
University of Waterloo
brecht@cs.uwaterloo.ca

ABSTRACT

Despite the ubiquity of WiFi devices, Bluetooth is widely used for communication in low-power, low data-rate devices. This is because Bluetooth consumes much less power than WiFi which results in longer battery life. The higher power consumption of WiFi devices is due to overheads from either establishing or maintaining connections with the access point. Surprisingly, Bluetooth devices require nearly three times as much energy to transmit a bit of data *at the physical layer* than WiFi devices.

In this paper, we propose Wi-LE a WiFi-compatible communication system that avoids the power hungry process of establishing or maintaining a connection. We implement and evaluate Wi-LE using an off-the-shelf WiFi module. Our results show that Wi-LE has power consumption similar to that of Bluetooth Low Energy (BLE). This demonstrates the potential for Wi-LE to be used in place of BLE.

ACM Reference Format:
Ali Abedi, Omid Abari, and Tim Brecht. 2019. Wi-LE: Can WiFi Replace Bluetooth?. In *HotNets '19: ACM Workshop on Hot Topics in Networks, November 13–15, 2019, Princeton, NJ, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3365609.3365853

1 INTRODUCTION

Despite the ubiquity of WiFi devices, Bluetooth is still widely used for communication in many battery-powered devices such a smart watches, wireless headsets, and IoT sensors. The reason for this is that Bluetooth communication consumes orders of magnitude less power than WiFi communication and hence it enables much longer battery life.

HotNets '19, November 13–15, 2019, Princeton, NJ, USA

ACM ISBN 978-1-4503-7020-2/19/11…$15.00
https://doi.org/10.1145/3365609.3365853

Interestingly, although Bluetooth requires less energy to send one bit of information *at the application layer*, it actually requires nearly three times as much energy as WiFi to transmit a bit of data *at the physical layer*. At the physical layer the energy required to transmit one bit of data using Bluetooth is 275-300 nJ/bit [12, 14] while with WiFi it is 10-100 [10, 13] depending on the bitrate. WiFi devices require less energy at the physical layer because they use much more efficient modulation techniques than Bluetooth. Unfortunately, MAC-layer overheads in WiFi, such as establishing and maintaining a connection with the access point, significantly increase the total power consumption. Due to these overheads, Bluetooth outperforms WiFi in terms of total power consumption.

The problem is that for applications requiring only periodic WiFi communication with small amounts of data (e.g., IoT devices), considerable overheads are incurred to establish and maintain the connection to the access point. For example, consider a battery-powered wireless temperature sensor which spends most of its time in sleep mode to save energy but periodically wakes up (e.g., every 10 minutes) to send its temperature reading to a base station. If this sensor utilizes a WiFi radio, upon waking from sleep mode, it must reassociate with the base station before transmitting data. As a result, in order to transmit a few bytes of actual data, many management frames including probe request/response and association request/response frames have to be exchanged. This consumes a significant amount of energy while the device establishes a connection and contributes to a relatively high amount of energy being consumed.

Given the fact that at the physical layer WiFi is fundamentally more energy efficient than Bluetooth, this paper examines possibilities for lowering the overall power consumption of WiFi in order to enable *WiFi-compatible* communication for IoT devices that only require intermittent communication. The goal is to devise techniques that would enable a low data-rate IoT device to communicate using WiFi with power consumption that is comparable to that of Bluetooth Low Energy (BLE).

In this paper, we conduct an in depth study of where and why so much energy is consumed when communicating using WiFi devices and examine several different approaches to reducing overall power consumption. We propose *WiFi Low*

Energy (Wi-LE), a novel approach that enables WiFi communication using power consumption levels that rival Bluetooth Low Energy (BLE). Wi-LE eliminates several overheads incurred by the 802.11 protocol while maintaining WiFi compatibility. This enables low-power IoT devices to periodically transmit their data without the need for re-association while still allowing the devices to enter power-saving sleep mode. Wi-LE achieves this without any change in the physical layer or MAC layer, and it works with existing WiFi devices.

Low power WiFi communication provides significant advantages over BLE such as: easy integration with the growing number of ubiquitous WiFi devices, reduced costs for manufacturers by potentially eliminating the need for Bluetooth functionality in some devices, and enabling the use of the 5 GHz spectrum (allowing devices to avoid the increasingly crowded 2.4 GHz spectrum used by BLE).

The key contributions of this paper are:

- We propose and develop Wi-LE, a WiFi-based communication system that has similar power requirements and obtains data rates comparable with Bluetooth Low Energy (BLE).
- We implement a prototype of Wi-LE using an off-the-shelf micro-controller that includes integrated WiFi. We characterize the power consumption of Wi-LE and compare it with other approaches to reducing power consumption of WiFi devices (e.g., using different power saving modes). Our results show that Wi-LE achieves energy efficiency of 84 μJ per message while the best alternative WiFi approach achieves 19.8 mJ per message.
- Wi-LE enables a wider range of deployment scenarios than previously possible with either Bluetooth or WiFi. For example, when available, Wi-LE can utilize existing WiFi infrastructure (which Bluetooth cannot), or in environments with no WiFi infrastructure such as farms Wi-LE enables wireless communication directly between IoT devices and a WiFi device such as a smartphone.

2 RELATED WORK

WiFi-based backscatter technology has recently been proposed to enable low-power WiFi communications [3, 19–21]. Although the low power consumption of these technologies is very attractive, they suffer from multiple practical issues. First, the range of these systems is very limited. In order to work, the backscatter devices have to be placed very close (i.e., within a meter) to the WiFi transmitter or receiver. Second, these systems require software or hardware modification to WiFi access points and devices. Finally, they require at least two WiFi devices to be able to operate. In contrast to backscatter technologies, Wi-LE uses active transmission and is different in three key aspects: first, the range of Wi-LE is much higher than WiFi-based backscatter systems. In fact, the range of Wi-LE is the same as typical WiFi. Second, Wi-LE does not require two WiFi devices to operate. A single WiFi device or an access point is enough for Wi-LE to communicate. Finally, since Wi-LE can be implemented using off-the-shelf devices, and is compliant with WiFi standard, it does not require any modifications to existing WiFi devices.

The work closest to ours is a technique called WiFi beacon-stuffing [6, 18]. This technique overloads some fields in the 802.11 beacon and other management frames with data containing information such as location-specific advertisements. Specifically, a WiFi access point embeds additional data into its beacons for the purpose of multi-casting information to nearby devices. In contrast with that work, Wi-LE injects WiFi beacons to eliminate the need for the power-hungry re-association process and hence enables low power WiFi communication for low data-rate IoT devices.

One of the advantages of Wi-LE is that it does not require any WiFi infrastructure to operate. This feature is also present in *WiFi direct* [4, 17], which enables WiFi devices to communicate directly with each other without an access point. However, in a fashion similar to that used in infrastructure-based WiFi networks, WiFi direct requires exchanging several management frames to establish a connection. Despite the power management schemes of WiFi direct, its power consumption is similar to that of WiFi [5].

Similarly, WiFi ad-hoc networks do not require any infrastructure to operate. Empirical measurements have shown that ad-hoc WiFi communication consumes even more power than when using infrastructure mode [8, 9]. This is mainly because overheads are still incurred to establish and maintain an ad-hoc connection. Moreover, ad-hoc nodes are additionally responsible for some tasks that are otherwise performed by an access point when operating using infrastructure mode. Examples of such tasks are generating beacons and handling time synchronization between nodes.

3 BACKGROUND

3.1 Establishing an 802.11 Connection

In this section, we provide a brief description of the required steps for establishing a connection with a WiFi access point. First, a WiFi client device requires some information about the access point before attempting to connect. The client either needs to wait to receive a beacon frame from the AP or actively probe the AP by sending a probe request to the AP. The AP responds by sending a probe response. The probe response includes information about the capabilities of the AP such as supported transmission rates.

In the next step, the client transmits an authentication request to the AP. Upon receiving the authentication request,

the AP first sends an ACK and then sends an authentication frame to the client to indicate successful authentication. The client then acknowledges the reception of the authentication frame. Next, the client and the AP exchange association request and response frames. If the access point has encryption enabled, another step is required to validate the shared key. In this example, we assume that the AP uses 802.1x authentication [1] which is the case for the Google WiFi AP [2] we utilize in our tests. A four-way handshake is performed using the 802.1x protocol to confirm that the client has the shared-key. At least 8 frames are exchanged during this process.

In addition to these 20 MAC-layer frames, 7 higher-layer frames including DHCP and ARP have to be transmitted before a client device can transmit to the AP. Transmitting all of these frames consumes a significant amount of power in order to establish a WiFi connection.

3.2 Maintaining an 802.11 Connection

Maintaining an 802.11 connection is also a power-hungry process. A client has to listen on the wireless channel to receive packets from the AP. Otherwise, the AP concludes that the client has disconnected. Unfortunately, the WiFi radio consumes a lot of power even when not transmitting or receiving packets. To alleviate this problem the 802.11 standard incorporates a power saving mechanism that enables clients to significantly reduce their power consumption while staying connected to the AP.

At a high level, while using this power saving mechanism, a client turns off its radio when it has no packets to transmit and only wakes up periodically to receive the beacon frames transmitted by the AP. Since the transmission of the beacons is periodic, a client can accurately calculate when to wake up. The access point indicates in the beacon if it has any packet for each connected client. If a client finds out that there are packets queued for it at the AP, it then asks the AP to transmit the packets, otherwise it goes back to sleep. This procedure significantly reduces the power consumption of the device while in the idle mode. Despite this significant reduction in power consumption, the cost of maintaining a WiFi connection is still extremely high for a battery-operated IoT device. In Section 5, we show how the power draw while in power-saving mode impacts the overall power consumption.

4 Wi-LE

Wi-LE is a WiFi-based communication system for low data rate IoT applications. It significantly reduces the power consumption of WiFi by taking a different approach to the communication protocol. In WiFi networks, establishing a connection to an access point involves multiple steps including authentication, association, and encryption. These steps are required for network management reasons and to prevent unauthorized users from accessing services such as connecting to the Internet. These steps impose extra overhead on the WiFi protocol by exchanging many messages at the time a connection is established which consequently increases the power consumption. However, an IoT device that provides a service (e.g., reporting the temperature) may not need to consume the energy required to establish a connection.

In Wi-LE, an IoT device broadcasts WiFi packets to send its data without joining any WiFi network. This feature is called "packet injection" and is supported by many WiFi chipsets. Nearby WiFi devices such as smartphones can receive the injected packet. However, since this packet does not belong to the WiFi network that the smartphone is connected to, it will be dropped at the MAC layer, unless the WiFi chip is in the monitor mode. Unfortunately, monitor mode is not supported by all WiFi cards. For smartphones and tablets running Android or iOS, it also requires rooting the device in order to put the WiFi card into monitor mode.

We solve this problem by injecting fake WiFi beacon frames that carry the IoT device's data. In other words, the IoT device pretends to be an access point. This beacon frame is received by all nearby WiFi devices. Upon receiving a WiFi beacon frame, the MAC layer forwards it to higher layer to notify the operating system and thereby the user about the existence of the WiFi network. Operating systems typically use this information to show a list of APs (SSIDs) the user can connect to. Therefore, an IoT device can transmit its data to nearby WiFi devices by injecting WiFi beacon frames. For instance, Figure 1 shows a temperature sensor that embeds its data in 802.11 beacon frames. The advantage of this approach is that it requires no software or hardware modifications (e.g., rooting the phone) on the receiving device which can be a smartphone, a tablet, or a computer. Instead a simple Android or iOS application or other software running on a host can retrieve the sensor's data. This application looks for special beacon frames transmitted by IoT devices and extracts their data from the beacon frames.

4.1 How to Avoid Spamming

If IoT devices are in the vicinity of another WiFi device their transmitted data would make it appear as an access point to that WiFi device (potentially spamming the list of access points available to that device). Users would see a long list of fake access points on their phones or computers which can adversely impact the user experience. To avoid this problem, Wi-LE utilizes the "hidden SSID" mechanism in the 802.11 standard. The hidden SSID feature allows users to hide their WiFi networks by not advertising the name of their SSID. As a result, the access point is not shown on the list of available

Figure 1: In Wi-LE, an IoT device sends out fake WiFi beacon frames. All nearby WiFi devices receive these beacons and can extract the IoT device's data.

WiFi networks. Since the SSID field must be null in the hidden SSID mode, Wi-LE must place IoT devices' data in other fields. The "vendor specific" information element field in the 802.11 beacon frame is a suitable place for our purposes. This field can be up to 253 bytes [6] and does not have any specific format and can therefore be used to transmit an string.

In Wi-LE, an IoT device inserts its data into the vendor specific field of a fake 802.11 beacon frame and broadcasts it using frame injection. Using this technique nearby WiFi devices to not see any additional access points in their list of available WiFi networks. Therefore, Wi-LE does not interfere with the normal operation of WiFi networks. However, the beacons transmitted by IoT devices can be retrieved from the operating system.

By utilizing the beacon injection technique, Wi-LE avoids the overheads of establishing and maintaining a WiFi connection. The microcontroller can enter sleep mode between the periodic transmissions. When the microcontroller wakes up, it embeds its data in a beacon frame, transmits it immediately and goes back to sleep. Note that Wi-LE does not associate with an AP for transmission.

5 EVALUATION

5.1 Experiment Setup

We use a Google WiFi access point [2] and an off-the-shelf ESP32 WiFi/BLE [7] as an IoT device to perform our experiments [1]. This module is a low-cost and low-power system-on-chip microcontroller with integrated Bluetooth and WiFi chips. The Bluetooth chip supports Bluetooth v4.2 BR/EDR and BLE. The WiFi chip operates at 2.4 GHz and supports the IEEE 802.11 b/g/n standards. The chip also supports WiFi packet injection which is critical for the implementation of Wi-LE. The module's microcontroller is an Xtensa dual-core 32-bit LX6 microprocessor, operating at up to 240 MHz. To reduce power consumption, we set the default frequency to 80 MHz which is the lowest frequency required for WiFi and Bluetooth functionality. The microcontroller also provides a few power saving modes namely, deep sleep, light sleep, and automatic light sleep, that are critical for implementing ultra-low-power systems. In deep sleep mode, the CPU and RAM are disabled and only a timer is active to wake up the microcontroller. The current draw in deep sleep mode is as low as 2.5 μA. In light sleep mode, we have full RAM retention so the wake up procedure is much faster than when in deep sleep mode at the cost of higher power consumption. The current draw during light sleep mode can be as low as 0.8 mA. The WiFi radio is disabled in both light and deep sleep modes. The ESP32 also supports an automatic light sleep mode in which the WiFi radio and microcontroller go to sleep in between WiFi beacon frames and wake up only to receive beacon frames in order to maintain the connection. The current draw while in automatic light sleep mode with active WiFi is about 5 mA.

As illustrated in Figure 2, we utilize a Keysight 34465A digital multimeter [11] to measure the current draw from the ESP 32 WiFi module. This multimeter is capable of taking 50,000 samples per second with pico ampere (pA) accuracy which enables us to measure the power consumption very accurately. To measure the total current draw by ESP32 module, we place the multimeter in series with the 3.3 volt DC power source and the module.

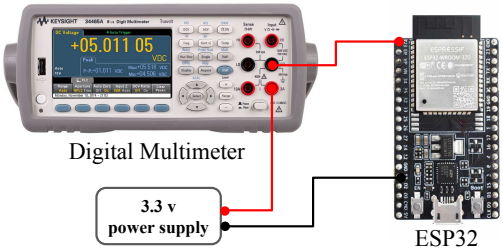


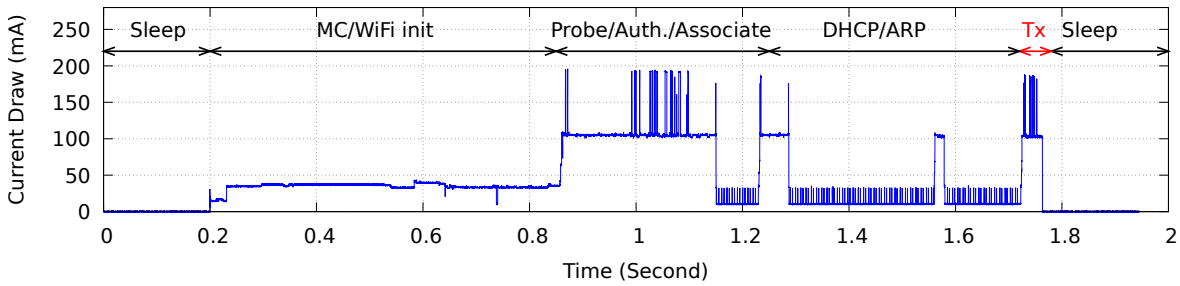
Figure 2: Experiment setup

5.2 Understanding Wi-LE Operation

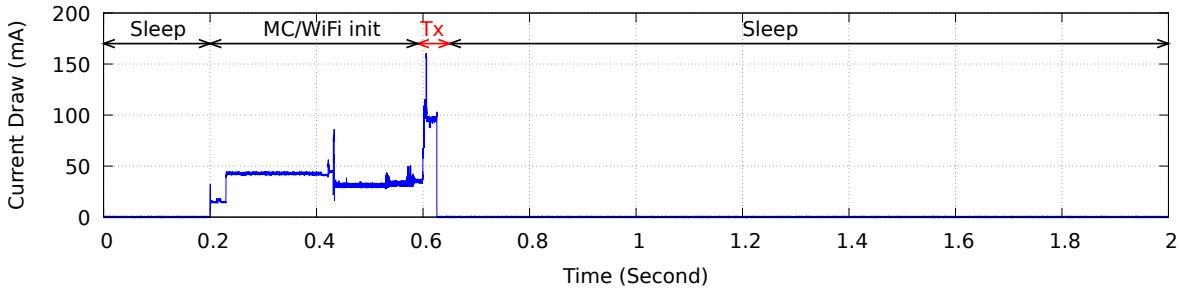
In this section, we examine the power consumption required to establish an 802.11 connection and compare that value with Wi-LE's connection-less approach. As described in Section 3.1, a WiFi client device has to exchange many MAC and network layer messages with the WiFi access point in order to establish a WiFi connection. Establishing the connection is a necessary step in today's WiFi networks in order to transmit a data packet.

Figure 3a shows the current consumed by the ESP32 WiFi module when transmitting a data packet after waking up from deep sleep mode. The ESP32 disables the WiFi radio in deep sleep mode to save energy. Therefore, it has to establish

[1] To eliminate unnecessary power draw on the ESP32 evaluation PCB board, we removed the voltage regulator and LED from the board and provide a clean 3.3 volt DC source of power directly from a power supply.



(a) WiFi



(b) Wi-LE

Figure 3: The current consumed by WiFi and Wi-LE for transmitting a frame

the WiFi connection after coming out of sleep mode. In the first phase (i.e., from time 0.2 to 0.85 s), the microcontroller executes the start up procedure which involves reading from the persistent storage (i.e., flash). It then initializes the WiFi module to prepare it for connecting to an AP. In the second phase (i.e., from time 0.85 to 1.15 s), the WiFi module exchanges the MAC management frames such as probe, authentication, association request and response frames. It also exchanges several messages to pass the WPA2 encryption mechanism. The spikes we observe in this phase correspond to transmission and reception of these management frames.

In the next phase, DHCP and ARP messages are exchanged so that the client device receives an IP address and finds the MAC address of the destination (i.e., the AP) for the data packet it wants to transmit. As you can see in the figure, the current draw drops to 20-30 mA for most of this phase. We have enabled Dynamic Frequency Scaling (DFS) and automatic light sleep as described in Section 5.1 to reduce the transmission power when the microcontroller and the WiFi module are idle. As can be seen in the figure there are fairly long wait times for network layer messages such as DHCP, therefore the modules enters its power saving mode. Finally, the WiFi module becomes ready to transmit the data packet. It only takes a few milliseconds for the data transmission to finish as depicted by red arrow in Figure 3a. After the transmission, the microcontroller re-enters the deep sleep mode.

Figure 3b shows the current consumed by Wi-LE for transmitting a data packet. Similar to the WiFi experiment, in the first phase the microcontroller and the WiFi module have to be initialized. Interestingly this step is shorter when compared with the WiFi case. This is because of a simpler initialization phase for Wi-LE. Specifically, in Wi-LE, the chip does not need to prepare to connect to the AP as a client; it can simply enable the WiFi radio to inject a packet without any association. Finally, the WiFi module broadcasts the beacon frame as described in Section 4 and goes back to the sleep mode. These figures clearly show that Wi-LE significantly reduces the total time and energy required to transmit a packet.

5.3 Scenarios Studied

We study different WiFi and BLE communication scenarios to compare the efficacy of Wi-LE in reducing the power consumption of WiFi.

WiFi Power Saving (WiFi-PS): In this scenario, the WiFi chip associates with an access point and maintains the connection by utilizing aggressive power saving mode. In this mode, during the idle time, the WiFi device skips receiving some beacon frames. Specifically, the WiFi chip wakes up

only for every third beacon frame. Finally, the microcontroller is in the automatic light sleep mode which reduces its clock frequency and utilizes clock gating to reduce the power consumption considerably.

WiFi Duty Cycle (WiFi-DC): In this scenario, the WiFi chip disconnects from the AP after transmitting its data and goes to sleep to save power. The microcontroller goes to the deep sleep mode to reduce the power consumption to the lowest level possible. A timer is set to wake up the microcontroller for the next transmission. The WiFi device has to re-associate with the AP before its next transmission.

Bluetooth Low Energy (BLE): In this scenario, the BLE chip is in the slave mode, and periodically transmits a data packet to another BLE device which is in the master mode. The microcontroller goes into the deep sleep mode between the transmissions.

Wi-LE: In this scenario, the WiFi chip injects a beacon frame without associating with any access point. The AP (i.e. another WiFi card) is in the monitor mode to receive and verify these beacon frames. The microcontroller goes into the deep sleep mode between the transmissions. Note that BLE and Wi-LE are compared under identical scenarios. Specifically, both techniques periodically transmit a packet (i.e., one-way communication) and spend the rest of the time in sleep mode.

5.4 Energy per Packet

In this section, we study the energy to transmit a data packet using different technologies. This is an important metric because each time the IoT device wants to transmit, it needs to sent at least one message regardless of how small its data is. To calculate the required energy per packet, we measure the time the microcontroller and WiFi module are on while transmitting a packet. We also measure the average power consumption during this time. We then multiply these numbers to calculate the energy. We also measure the current consumed while in idle mode (i.e., in-between transmissions). Table 1 summaries our measurements for all scenarios studied.

We consider two WiFi modes: 1) the WiFi client always stays connected to the AP and it is in power saving mode (WiFi-PS) 2) the WiFi client disconnects from the AP and goes into deep sleep mode in between transmissions (WiFi-DC). Table 1 shows that when the client stays connected to the AP (WiFi-PS) the energy it requires to transmit a packet is an order of magnitude smaller than when the client needs to re-associate. On the other hand, the idle current consummation is about 2000 times more in WiFi-PS since the microcontroller is in the automatic light sleep mode instead of the deep sleep mode.

Although our ESP32 module supports BLE, we do not use it as our reference for BLE power consumption because their Bluetooth implementation is inefficient in terms of power consumption (i.e., it is close to WiFi power consumption) and still under development. Instead, we use a CC2541 [16] which is an ultra-low power BLE module as our reference for power consumption. Table 1 presents the power consumption results from a report [15] published by the chipset's manufacturer. We observe that the energy per packet for BLE is almost three orders of magnitude lower than WiFi-PS. This is why BLE modules can run on a small button battery for over a year.

We previously observed that in Wi-LE, the transmission time of a packet is mostly spent waiting for the microcontroller and the WiFi card to become ready before it can transmit a packet. These steps are unavoidable for regular WiFi because it requires powerful and complex microcontrollers and WiFi modules to implement the full protocol stack in order for WiFi to operate. On the other hand, Wi-LE does not require the entire protocol stack since it only broadcasts an 802.11 beacon frame without any connection. The content of the packet including all of headers can be pre-computed and then only the IoT device's data needs to be inserted into the packet. As a result, Wi-LE can be implemented very efficiently in hardware to significantly reduce the initialization and setup time. To compute the energy per packet for Wi-LE in Table 1, we consider only the time required to transmit the packet and multiply that by the power consumption measured from the ESP32 modules. We find that Wi-LE's energy per packet is 84 μJ which is very close to that of BLE. For this measurement, we use a physical bitrate of 72 Mbps at transmission power of 0 dBm which has a similar range as BLE at the same transmission power (i.e., a few meters). We believe that an application-specific integrated circuit (ASIC) implementation will have much lower power consumption.

	Wi-LE	BLE	WiFi-DC	WiFi-PS
Energy/packet	84 μJ	71 μJ	238.2 mJ	19.8 mJ
Idle current	2.5 μA	1.1 μA	2.5 μA	4500 μA

Table 1: Energy required to transmit a message using different technologies and their idle current comparison

5.5 Average Power Consumption

An important factor for low-power systems is the average power consumption. We use the following formula to compute the average power consumption:

$$P_{avg} = \frac{1}{INT}\left(\frac{P_{tx}}{T_{tx}} + \frac{P_{idle}}{INT - T_{tx}}\right) \quad (1)$$

where P_{tx} and P_{idle} are the power consumption in transmission and idle mode. T_{tx} is the duration of transmission including all overheads such as the initialization of the microcontroller. The interval between transmissions is denoted by INT.

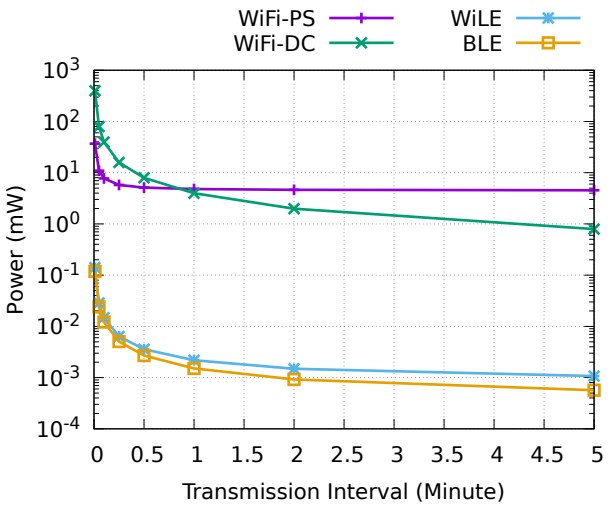


Figure 4: The comparison of overall power consumption for different transmission intervals

Figure 4 shows the average power consumption for different transmission intervals (i.e., INT). The average power consumption generally decreases as we increase the interval between transmission. This is because fewer power hungry transmissions happen over a given period of time. We observe that if a device transmits its data more than once per minute WiFi-PS outperforms WiFi-DC due to the high overhead of association in WiFi-DC. However, if the transmission period is longer, WiFi-DC performs better because the idle transmission power used in WiFi-PS outweighs the overhead of re-association. This figure also shows that the power consumption of Wi-LE is close to that of BLE and generally about 3 orders of magnitude lower than any of the WiFi solutions.

6 DISCUSSION

In this paper, we have investigated the possibility of replacing Bluetooth with WiFi. Our preliminary results show that Wi-LE is a promising technique that enables WiFi communication at power consumption levels that rival Bluetooth Low Energy (BLE). Nevertheless, in order to enable a full communication system the following topics require further study:

Network of IoT devices: We implement and evaluate Wi-LE for one IoT device. Extending this work to a network of IoT devices is the subject of future work. The messages generated by IoT devices must contain unique identifiers so that they can be distinguished from each other. The possibility of concurrent transmissions from multiple devices and the mitigation mechanism need to be studied. We believe that if two devices happen to transmit at the same time and they have the same transmission period, their transmissions will automatically differ away from each other due to the jitter of their clocks.

Two-way communication: In this paper, we focus on the transmission of data from an IoT device to nearby WiFi devices. Although this is sufficient for many applications that require one-way communication, it would be ideal if Wi-LE supports two-way communication. The challenge of receiving WiFi packets efficiently is that the receiver needs to actively wait for packets and this is a power hungry process. One way to solve this challenge is that the IoT device specifies when the packets destined to the device should be transmitted. For instance, an IoT device that utilizes Wi-LE can indicate in some beacon frames that it will be ready to receive packets for a short time slot after the current beacon. This way the waiting period will be limited to the time slots specified by the IoT device and therefore the power consumption is reduced significantly.

Security: Currently, since Wi-LE systems communicate by injecting raw packets with no encryption all devices within range of the sender can obtain the transmitted data by monitoring the channel. However, security can be easily provided by encrypting the data prior to its transmission.

7 CONCLUSION

In this paper, we present Wi-LE a low-power WiFi communication system that avoids the overheads of establishing and maintaining a WiFi connection. Instead, it injects 802.11 beacon frames that can be received by all nearby WiFi devices. Wi-LE is suitable for IoT applications where a device needs to periodically transmit some data to a base station or a smartphone. Our evaluations show that the power consumption of Wi-LE is similar to that of Bluetooth low energy.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful feedback. We also thank the Natural Sciences and Engineering Council of Canada (NSERC) for partial funding for this project.

REFERENCES

[1] 2010. *IEEE 802.1X-2010.* https://standards.ieee.org/standard/802_1X-2010.html.
[2] 2019. *Google Wifi.* https://store.google.com/product/google_wifi.

[3] Ali Abedi, Mohammad Hossein Mazaheri, Omid Abari, and Tim Brecht. 2018. WiTAG: Rethinking Backscatter Communication for WiFi Networks. In *HotNets*. 148–154.

[4] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano. 2013. Device-to-device communications with Wi-Fi Direct: overview and experimentation. *IEEE Wireless Communications* 20, 3 (2013), 96–104.

[5] Daniel Camps-Mur, Xavier Pérez-Costa, and Sebastií Sallent-Ribes. 2011. Designing Energy Efficient Access Points with Wi-Fi Direct. *Comput. Netw.* 55, 13 (Sept. 2011), 2838–2855.

[6] R. Chandra, J. Padhye, L. Ravindranath, and A. Wolman. 2007. Beacon-Stuffing: Wi-Fi without Associations. In *Eighth IEEE Workshop on Mobile Computing Systems and Applications*. 53–57.

[7] Espressif Systems 2019. *ESP32 datasheet*. Espressif Systems. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[8] L. M. Feeney and M. Nilsson. 2001. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *INFOCOM*. 1548–1557.

[9] R. Friedman, A. Kogan, and Y. Krivolapov. 2013. On Power and Throughput Tradeoffs of WiFi and Bluetooth in Smartphones. *IEEE Transactions on Mobile Computing* 12, 7 (2013), 1363–1376.

[10] Daniel Halperin, Ben Greenstein, Anmol Sheth, and David Wetherall. 2010. Demystifying 802.11N Power Consumption. In *Proceedings of the 2010 International Conference on Power Aware Computing and Systems (HotPower'10)*.

[11] Keysight technologies [n. d.]. *34465A Digital Multimeter*. Keysight technologies. https://literature.cdn.keysight.com/litweb/pdf/5991-1983EN.pdf?id=2318052.

[12] Konstantin Mikhaylov, Nikolaos Plevritakis, and Jouni Tervonen. 2013. Performance Analysis and Comparison of Bluetooth Low Energy with IEEE 802.15.4 and SimpliciTI. *Journal of Sensor and Actuator Networks* 2 (2013), 589–613.

[13] S. K. Saha, P. Deshpande, P. P. Inamdar, R. K. Sheshadri, and D. Koutsonikolas. 2015. Power-throughput tradeoffs of 802.11n/ac in smartphones. In *2015 IEEE Conference on Computer Communications (INFOCOM)*. 100–108.

[14] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen. 2012. How low energy is bluetooth low energy? Comparative measurements with ZigBee/802.15.4. In *2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 232–237.

[15] Texas Instruments 2012. *Measuring Bluetooth Low Energy Power Consumption*. Texas Instruments. http://www.ti.com/lit/an/swra347a/swra347a.pdf.

[16] Texas Instruments 2013. *CC2541: 2.4-GHz Bluetooth low energy and Proprietary System-on-Chip*. Texas Instruments. http://www.ti.com/lit/ds/symlink/cc2541.pdf.

[17] WiFi Alliance [n. d.]. *Wi-F-Direct*. WiFi Alliance. https://www.wi-fi.org/discover-wi-fi/wi-fi-direct.

[18] S. Zehl, N. Karowski, A. Zubow, and A. Wolisz. 2016. LoWS: A complete Open Source solution for Wi-Fi beacon stuffing based Location-based Services. In *2016 9th IFIP Wireless and Mobile Networking Conference (WMNC)*. 25–32.

[19] Pengyu Zhang, Dinesh Bharadia, Kiran Joshi, and Sachin Katti. 2016. HitchHike: Practical Backscatter Using Commodity WiFi. In *SenSys*.

[20] Pengyu Zhang, Colleen Josephson, Dinesh Bharadia, and Sachin Katti. 2017. FreeRider: Backscatter Communication Using Commodity Radios. In *CoNEXT*.

[21] Jia Zhao, Wei Gong, and Jiangchuan Liu. 2018. Spatial Stream Backscater Using Commodity WiFi. In *MobiSys*.