Fix It, Don't Bin It! - CRC Error Correction in Bluetooth Low Energy

Evgeny Tsimbalo, Xenofon Fafoutis and Robert Piechocki
Department of Electrical and Electronic Engineering
University of Bristol
{e.tsimbalo, xenofon.fafoutis, r.j.piechocki}@bristol.ac.uk

***Abstract*—In this paper, we introduce error correction to the Bluetooth Low Energy (BLE) standard by utilising data redundancy provided by the Cyclic Redundancy Check (CRC) code used to detect erroneous packets. We assume a scenario with an energy-constrained transmitter and a constraint-free infrastructure, which allows us to introduce additional signal processing at the receiving side while keeping the transmitter intact. A novel approach of applying iterative decoding techniques to the BLE CRC code is investigated in this work. By using these techniques and real BLE packets collected in an office environment, we show that by enabling CRC error correction, the sensitivity of the BLE receiver can be improved by up to 3 dB. At the same time, up to 60% of corrupted packets can be corrected, which directly translates to a significant reduction in the number of retransmissions and a noticeable energy saving.**

***Keywords*-CRC; error correction; Bluetooth Low Energy**

I. Introduction

The Bluetooth Low Energy (BLE) communication protocol is a subset of the Bluetooth 4.x standard [1] that is aimed at very low power applications. As one of the *de facto* wireless communication options in modern smart phones, BLE has become a widely adopted choice for many manufacturers of commercial wearable gadgets. Moreover, BLE has been used beyond the limits of a personal area networks. In the context of a smart space, the iBeacon technology [2], which is based on BLE, enables proximity-based services and applications, such as indoors positioning. In smart homes and e-Health infrastructures, BLE has been used for the wireless links between wearable computers and the infrastructure [3]. All in all, BLE is considered as one of the key technologies to power up the Internet of Things (IoT).

In line with many communication protocols, BLE uses Cyclic Redundancy Check (CRC) codes to detect communication errors. Prior to transmission, the packet is processed by a CRC encoder, which generates 24 redundant bits (hence the name of the code, CRC-24) and appends them to the end of the packet. At the receiver side, the received packet is forwarded to the higher layers of stack only if a CRC check is successful, otherwise the packet is considered corrupted and is dropped.

In the application layer, corrupted packets are experienced as performance degradation, whose nature depends on which of the two BLE modes is used. More specifically, in BLE, a device can operate in any of four distinct roles, namely *Broadcaster*, *Observer*, *Peripheral*, and *Central*. The first two roles are based on a connectionless one-directional communication model where the *Broadcaster* is the transmitter and the *Observer* is the receiver. The last two roles are based on a connection-oriented model, where the *Central* is the master and the *Peripheral* is the slave. In the connection-oriented case, BLE offers link-layer acknowledgements and retransmissions. The receiver notifies the transmitter, via an acknowledgement, whether the packet needs to be retransmitted. Corrupted packets are experienced in the transmitter as an additional source of energy consumption. The more retransmissions are required for a BLE packet to pass through the link, the more energy is consumed. In the connectionless case, instead, corrupted packets are experienced as application-layer data loss.

While CRC codes are traditionally used for error detection only, they have inherent error correction potential due to redundancy they introduce to transmitted data. Regardless of which BLE mode is used, CRC error correction would directly improve the packet reception rate at the receiver and decrease the amount of energy that is spent on retransmissions at the transmitter. Considering that performing error correction requires additional processing at the receiver, this approach is essentially offering a way to decrease the number of retransmissions and unburden the transmitter by moving some of these costs to the receiver. Therefore, CRC error correction is a compelling choice for applications where an energy-constraint transmitter communicates with a constraint-free infrastructure, such as wearable sensors streaming data to a smart infrastructure.

In this paper, we investigate the error correction potential of the CRC-24 code in a real environment. Following our previous work that is based on simulation [4], we apply modern iterative decoding techniques to correct errors, while keeping the transmitter intact. To the best of our knowledge, these techniques have never been investigated in the context of BLE. The contribution of the paper can be summarised as follows. First, we use an off-the-shelf BLE kit to collect a dataset of real corrupted packets. We then implement and apply two state-of-the-art iterative decoding techniques to the dataset. For benchmarking, we also implement a simple look-up algorithm that is able to correct corrupted packets with no more than a single bit error. By comparing the techniques, we identify different situations where one of the techniques is more beneficial than the others.

The remainder of the paper is organised as follows. In

Section II we provide a brief overview of the background and related work. In Section III, we introduce two iterative decoding algorithms, namely Belief Propagation (BP) and Alternating Direction Method of Multipliers (ADMM), traditionally used in the context of Low Density Parity Check (LDPC) codes. In Section IV, we apply, evaluate and compare the performance of the decoding algorithms using a dataset of real corrupted BLE packets. Finally, Section V concludes the paper.

II. Background and Related Work

Some error correction techniques for CRC codes have been proposed over the years. A simple look-up algorithm correcting all single errors was described by the inventors of CRC codes in [5]. This algorithm is based on the fact that each possible single-error pattern gives a unique remainder after polynomial division of a corrupted packet by the generator polynomial, regardless of the actual bits transmitted in the packet. As a result, a look-up table of all possible remainders and the corresponding error bit positions can be calculated in advance, using packets with one in error positions and zeros elsewhere. When an erroneous packet with a single bit error arrives, the corresponding remainder is calculated and checked against the loop-up table, thus the error position is identified. This method can correct 100% of all single-error packets.

More sophisticated look-up techniques correcting some of double-error codewords were also developed [6]. However, all these techniques aim at correcting a particular number of errors. To the best of knowledge of the authors, no unified approach has been proposed to correct an arbitrary number of errors, limited only by the error-correction capabilities of the code itself.

Many state-of-the-art error correction codes employ iterative decoding algorithms. One of those algorithms, known as Belief Propagation (BP), was originally developed for Low Density Parity Check (LDPC) codes [7], a special type of linear codes that have a sparse parity check matrix. In general, BP provides good decoding performance when applied to any linear code, as long as the parity check matrix of the code is sparse and the corresponding Tanner graph contains no cycles of length four [8]. As shown in [9], the total number of four-cycles for an $m \times n$ parity check matrix $\mathbf{H}$ can be calculated as

$$\sum_{i=1}^{m} \sum_{j=i+1}^{m} \binom{(\mathbf{H}\mathbf{H}^T)_{ij}}{2}. \tag{1}$$

Based on (1), the number of four-cycles for the BLE CRC-24 code is 813816, which makes the direct application of iterative decoding techniques unfeasible.

In [9] and [8], the authors proposed methods to eliminate four-cycles on the Tanner graph of any linear code, demonstrating the results on Hamming and Reed-Solomon codes. The same techniques can be applied to CRC codes, making them suitable for BP-based error correction. For example, as shown in [4], when the maximum cycle strategy (MCS) algorithm [8] is applied to the BLE CRC-24 code, it produces an equivalent sparse version of the parity check matrix, while keeping all parity check equations intact. The Tanner graph of this modified matrix does not have any cycles of length four and therefore iterative decoding techniques can now be applied.

As an alternative to BP, the decoding of a linear code can be viewed as a linear program (LP), the idea that was first introduced in [10]. This approach resulted in an algorithm based on the Alternating Direction Method of Multipliers (ADMM) initially proposed in [11] and applied to LDPC codes in [12]. Practical and computationally simple modifications were further developed in [13], [14] and [15]. While the ADMM-based algorithm has only been investigated in the context of LDPC codes, it can also be applied to correct errors for any linear code, such as the BLE CRC-24.

The brief overview of both BP and the ADMM is given in the next section.

III. Iterative Decoding Techniques

A. BP

BP decoding in the form of the sum-product algorithm is one of the standard approaches to the decoding problem. We will use the log-likelihood version of the algorithm that is widely presented in the literature, for example, in [16]. The input of the algorithm are the log-likelihood ratios (LLRs) of the bits to be decoded. For a binary symmetric channel (BSC) with the crossover probability p, which can be viewed as an equivalent model of a CRC-based transmission, the LLR corresponding to the i-th received bit r_i can be calculated as

$$\gamma_i = (2r_i - 1) \ln \left(\frac{1-p}{p} \right). \tag{2}$$

For real channel environments, the crossover, or bit flipping, probability p can be estimated by transmitting known packets and calculating the bit error rate (BER) for each given received signal level. For auxiliary bits added by the sparsification process, the LLRs are set to zeros. After each iteration, hard decisions are made on the vector of updated LLRs and the syndrome is calculated using the original, non-sparse parity check matrix neglecting the auxiliary bits. If the syndrome is a zero vector, the original packet bits are extracted and the algorithm stops. The algorithm also stops if a maximum number of iterations is reached.

B. ADMM

Let $\mathbf{x} \in \{0,1\}^N$ denote the transmitted codeword corresponding to a CRC-encoded packet, and let $\mathbf{r} \in \{0,1\}^N$ denote the vector of received bits before the CRC check. We use $\mathbf{H}$ to denote the parity check matrix of the code, which, depending on the context, can be either the original, non-sparse matrix of the CRC-24 code or its sparsified version. In the case when it is the original matrix of size $M \times N$, the relationships between the CRC-encoded bits can be written as

$$\mathbf{H}\mathbf{x} = \mathbf{0}. \tag{3}$$

The equations (3) are a set of M constraints, also called checks, introduced by the CRC encoder between the bits. Let

c_i, $i = 1, .., N$, be the number of times the i-th bit participates in all checks, and let d_j, $j = 1, ..., M$ be the number of bits participating in the j-th check. The ADMM algorithm in its penalised form (ADMM-PD) introduced in [13] solves the following optimisation problem equivalent to the decoding of the received codeword:

$$\bar{\gamma}^T \mathbf{x} - \alpha \left\| \mathbf{x} - 0.5 \right\|_2^2, \tag{4}$$

$$\text{subject to} \forall j, \, \mathbf{P}_j \mathbf{x} = \mathbf{z}_j,$$

$$\mathbf{z}_j \in \mathbb{PP}_{d_j}. \tag{5}$$

Here, $\bar{\gamma} \triangleq -\gamma$ is a vector of negative LLRs; $\mathbf{P}_j$ is the operation of selecting those bits of $\mathbf{x}$ that participate in the j-th check; $\mathbf{z}_j$ is a replica vector for the j-th check; $\mathbb{PP}_{d_j}$ is the parity polytope of dimension d_j [12]; $\alpha > 0$ is a penalty coefficient. The l_2 penalty function in (4) was shown to provide the best PER performance in [13].

In line with our previous work [4], we adopt the over-relaxation technique advocated in [11] to improve decoding convergence. Denoting $\rho > 1$ the over-relaxation parameter, the ADMM-PD algorithm with with the l_2 penalty function is summarised in Algorithm 1. In the update for $\mathbf{z}_j$ in Algorithm 1, $\Pi_{\mathbb{PP}_{d_j}}(\cdot)$ is the projection onto the parity polytope $\mathbb{PP}_{d_j}$ [12]. In our implementation, we use the original projection algorithm proposed by [12]. We note that more computationally effective techniques were derived in [14] and [15].

As discussed in [4], the selected ADMM algorithm has several parameters: the augmented Lagrangian parameter μ, the penalty coefficient α, the over-relaxation parameter ρ and the maximum number of iterations T_{max}. Investigation into the selection of these parameters for some LDPC codes and the AWGN channel was carried out in [13], where it was shown that the algorithm is rather sensitive to parameters settings. In [4], for the BSC channel we obtained slightly different parameters values from the ones provided in [13] resulting in the following recommendations:

- $\mu \in [3, 5]$ provide the best error correction performance;
- The optimum penalty coefficient $\alpha = 1$;
- $\rho = 1.8$ provides the best performance and convergence;
- Increasing T_{max} improves the probability of error correction, at the expense of decoding speed; up to 1000 iterations can be considered.

To be consistent with our previous work based on simulation, we will use the values above to obtain practical performance results too. We note, however, that the optimum values of the ADMM parameters may be different depending on real channel conditions.

Algorithm 1 Iterative ADMM-PD algorithm with over-relaxation.

```
Input: Vector of negative LLRs $\bar{\gamma}$ and parity check matrix $\mathbf{H}$.
Output: Decoded vector $\mathbf{x}$.
1: Initialisation: Construct the selection matrix $\mathbf{P}_j$ for each check node $j$ based on $\mathbf{H}$. Initialise $\lambda_j$ as the all zeros vector and $\mathbf{z}_j$ as the all 0.5 vector.
2: Variable node update: For each variable node $i$, do:
    Calculate $\bar{\mathbf{z}}_j = \mathbf{P}_j^T \mathbf{z}_j^{[k]}$, $\bar{\lambda}_j = \mathbf{P}_j^T \lambda_j^{[k]}$, $\forall j$.
    Calculate $t_i = \sum_j \left( \bar{\mathbf{z}}_j - \frac{\bar{\lambda}_j}{\mu} \right) - \frac{\bar{\gamma}_i}{\mu}$.
    Update
        $x_i^{[k+1]} \leftarrow \frac{1}{c_i - 2\alpha_2/\mu}(t_i - \frac{\alpha_2}{\mu})$.
    Project $x_i^{[k+1]}$ onto $[0, 1]$: $x_i^{[k+1]} \leftarrow \Pi_{[0,1]} x_i^{[k+1]}$ .
3: Check node update: For each check node $j$, do:
    Calculate
        $\mathbf{v}_j^{[k+1]} \leftarrow \rho \mathbf{P}_j \mathbf{x}^{[k+1]} + (1 - \rho)\mathbf{z}_j^{[k]} + \frac{\lambda_j^{[k]}}{\mu}$.
    .
    Update $\mathbf{z}_j^{[k+1]} \leftarrow \Pi_{\mathbb{PP}_{d_j}}(\mathbf{v}_j^{[k+1]})$.
    Update
        $\lambda_j^{[k+1]} \leftarrow \lambda_j^{[k]} + \mu \left[ \rho \mathbf{P}_j \mathbf{x}^{[k+1]} + (1 - \rho)\mathbf{z}_j^{[k]} - \mathbf{z}_j^{[k+1]} \right]$ .
4: Make a tentative hard decision on $\mathbf{x}^{[k+1]}$: if $x_i^{[k+1]} \geq 0.5$, $\hat{x}_i = 1$; otherwise $\hat{x}_i = 0$.
5: If $\mathbf{H}\hat{\mathbf{x}} = \mathbf{0}$, then return $\mathbf{x} = \hat{\mathbf{x}}$. Otherwise, if $k + 1$ is smaller then the maximum number of iterations $T_{max}$, do $k \leftarrow k + 1$ and loop to Variable node update. Otherwise, declare decoding failure and Stop.
```

IV. Performance Evaluation

Aiming to investigate the performance of the aforementioned decoding techniques for real corrupted packets, our natural first step is to collect a dataset of packets with real channel errors. The corrupted packets were collected using a pair of nRF51822 kits. The transmitter is programmed as a *Broadcaster*, periodically transmitting advertisement packets at the maximum allowed frequency (10 Hz). We use the maximum allowed packet size (39 bytes), which results in the codeword length of 336 bits. We note that this is the worst-case scenario from the error correction point of view, due to the fact that it yields the smallest relative redundancy. The payload was populated with several copies of a single byte sequence number that is incremented by the transmitter. The receiver is programmed with the Nordic Sniffer 1.0.1, an application that monitors the BLE channels and provides the payload of the received packets in hex form, as well as additional information such as the Received Signal Strength Indicator (RSSI) and the result of the CRC check.

The dataset was collected in an office environment. The receiver was fixed to a computer, while the transmitter was positioned in different locations within the office and the surrounding areas. To minimise the number of corrupted packets due to interference, the data collection sessions were performed at off-peak hours when the office was empty. Furthermore, the transmitter was set to use BLE channel 39 located at the very end of the 2.4 GHz band, as opposed to WiFi channel 1 used by the local IEEE 802.11 access point. In

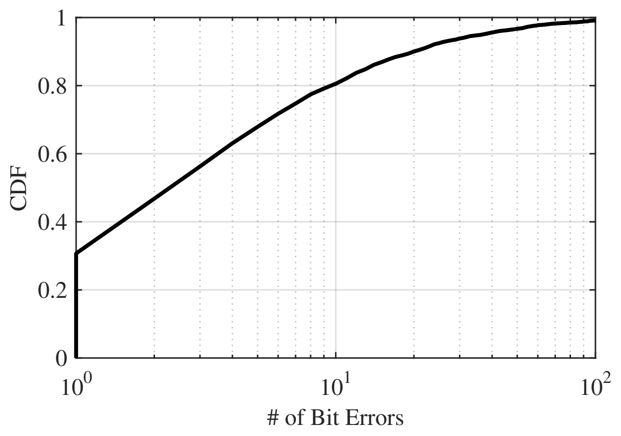

Fig. 1. CDF of the collected dataset of $6,000$ corrupted packets with respect to the number of bit errors per packet.

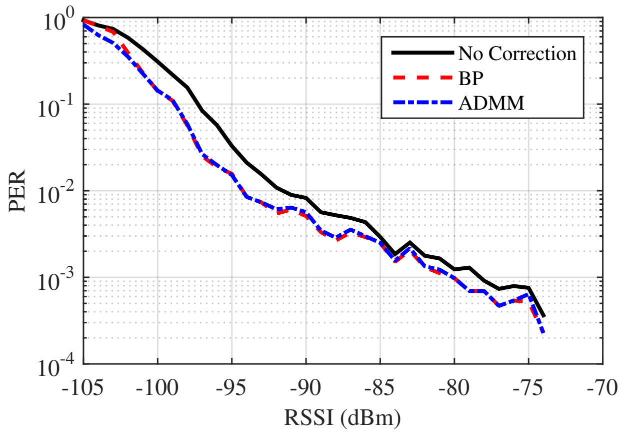

Fig. 2. PER for various RSSI levels before and after CRC correction techniques are applied.

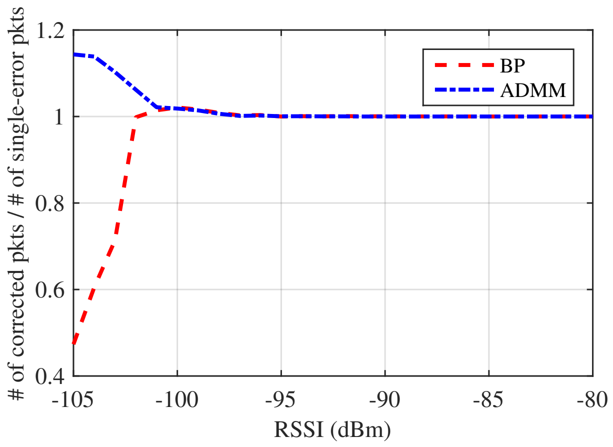

Fig. 3. Number of corrected packets for BP and ADMM relative to the number of single-error packets as a function of the RSSI.

a period of over 10 hours, we collected a total of approximately $400,000$ packets, $6,000$ of which were corrupted. Fig. 1 shows the CDF (Cumulative Distributed Function) of the corrupted packets with respect to the number of bit errors per packet. To better illustrate the occurrence of packets with a small number of errors, the horizontal axis is shown in the log scale. It can be observed that over 30% of the corrupted packets have a single bit error. Packets with multiple bit errors are increasingly more rare, with 80% of the erroneous packets having 10 bit errors or less. Packets that have not more than three errors make up the majority of all corrupted packets, indicating good error correction potential. We did not observe a significant difference in the bit error patterns at different RSSI levels.

Fig. 2 shows the Packet Error Rate (PER) for each RSSI level before any correction algorithm is applied (solid line). It can be seen that no packets are received at the RSSI of less than -105 dBm, which denotes the sensitivity threshold of the nRF51822 BLE receiver. Moreover, no corrupted packets are received at the RSSI of more than -74 dBm. We can observe that without error correction, the 1% PER threshold is at -91 dBm and the 0.1% PER threshold is at -78 dBm. On the other hand, when error correction is enabled, a gain of up to 3 dB can be achieved. In terms of the PER, it can be seen that both methods significantly improve the reliability, correcting up to 60% of corrupted packets at -97 dBm. While the ADMM-based algorithm outperforms BP in the region of low RSSI values (less than -102 dBm), both algorithms demonstrate similar performance at medium and high signal levels.

As mentioned in Section II, there exists a simple look-up algorithm that is able to correct all single error packets. To benchmark the performance of BP and ADMM to that of the look-up method, Fig. 3 illustrates the number of corrected packets in relation to the number of single-error packets as a function of the RSSI for both decoders. It can be observed that at RSSI levels of less than -100 dBm, BP would perform worse than the look-up method. By contrast, the ADMM would outperform the look-up method in this region, correcting up to 15% more packets. It can also be seen that all algorithms converge at RSSI values greater than -95 dBm. The reason is that when the received signal is strong, the PER is low and, therefore, any improvement in the number of corrected packets is relatively small. The performance comparison indicates that different methods could be used in different situations. Indeed, when the RSSI of the corrupted packet is greater than -95 dBm, the look-up method should be used, due to its simplicity and computational efficiency. When the RSSI is less than or equal to -95 dBm, the ADMM can correct more packets and, thus, results in fewer retransmissions.

In the connectionless mode, the benefits of using the CRC error correction algorithms are straightforward. As shown in Fig. 2, error correction can decrease the PER or, alternatively, increase the coverage. In the connection-oriented mode, on the other hand, the benefits are more subtle. As discussed in the introduction, CRC error correction reduces the number of

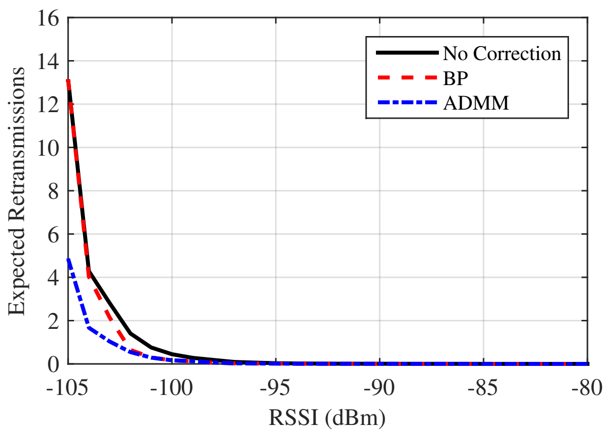

Fig. 4. Expected number of retransmissions with and without CRC error correction.

retransmissions, essentially decreasing the amount of energy consumed by the transmitter. To visualise this improvement in energy consumption, we model retransmissions as a sequence of independent Bernulli trials with the same probability of failure (i.e. PER) for each trial. The expected number of retransmissions can therefore be calculated as

$$\mathbb{E}\left[N\right] = \sum_{n=1}^{\infty} n(1-p)p^{n-1} - 1 = \frac{p}{1-p}, \qquad (6)$$

where p is the PER. The expected number of retransmissions for a given PER can be used as an approximation of the energy consumption of the transmitter normalised to the energy required to transmit a single packet. Fig. 4 illustrates the retransmission model given by (6) applied to the PER shown in Fig. 2. The results verify the previous findings, indicating a significant reduction in the number of retransmissions for RSSI levels lower than -95 dBm. For instance, at -102 dBm, the decoding algorithms decrease the required number of retransmissions from 1.5 to 0.5 with the corresponding energy saving equal to the energy required for a single packet transmission. In line with the previous figures, the ADMM outperforms BP by offering 60% fewer retransmissions when at the lowest RSSI level of -105 dBm.

V. Conclusions

In this paper, we analyse how the energy efficiency of the Bluetooth Low Energy transmitter can be increased by introducing error correction at the receiving side, without changing the existing CRC coding scheme. Our research is based on our previous work on CRC error correction, where simulations showed that modern iterative decoding techniques can be employed to correct errors. Using a pair of nRF51822 transceiver kits and an office environment, we first collected real erroneous BLE packets. After the decoding methods were applied, a gain of up to 3 dB in terms of the received signal level and the reduction in the PER of up to 60% was achieved. When compared with the look-up method allowing all single-error packets to be corrected, the ADMM-based algorithm demonstrated the best performance when the signal strength is low, correcting up to 15% more erroneous packets than the look-up method. Finally, error correction enabled significant reduction in the number of retransmissions, with the equivalent energy saving almost equal to the energy required for a single packet transmission. This can potentially extend the battery life of the transmitter or, alternatively, increase its range. We note that all these results were achieved without changing the BLE coding scheme, by just introducing additional signal processing at the receiver. Our future work includes complexity analysis and extension of the considered decoding methods to different packet lengths as well as other CRC codes.

Acknowledgment

This work was performed under the SPHERE (Sensor Platform for HEalthcare in a Residential Environment) IRC (Interdisciplinary Research Collaboration) funded by the UK Engineering and Physical Sciences Research Council (EPSRC), Grant EP/K031910/1.

References

[1] Bluetooth SIG, “Specification of the Bluetooth System - Covered Core Package version: 4.0,” [Online] Available at: https://www.bluetooth.org/en-us/specification/adopted-specifications, 2010.
[2] N. Newman, “Apple iBeacon technology briefing,” *J. Direct, Data Digit. Mark. Pract.*, vol. 15, no. 3, pp. 222–225, Jan. 2014.
[3] P. Woznowski, X. Fafoutis, T. Song, S. Hannuna, M. Camplani, L. Tao, A. Paiement, E. Mellios, M. Haghighi, N. Zhu, G. Hilton, D. Damen, T. Burghardt, M. Mirmehdi, R. Piechocki, D. Kaleshi, and I. Craddock, “A Multi-modal Sensor Infrastructure for Healthcare in a Residential Environment,” in *Proc. Int. Conf. Commun. (ICC) Workshops*, 2015.
[4] E. Tsimbalo, X. Fafoutis, and R. Piechocki, “CRC Error Correction for Energy-Constrained Transmission,” in *26th IEEE Int. Symp. Personal, Indoor and Mobile Radio Commun. (PIMRC)*, 2015.
[5] W. Peterson and D. Brown, “Cyclic Codes for Error Detection,” *Proc. IRE*, vol. 49, no. 1, pp. 228–235, Jan. 1961.
[6] S. Babaie, A. K. Zadeh, S. H. Es-hagi, and N. J. Navimipour, “Double Bits Error Correction Using CRC Method,” in *2009 Fifth Int. Conf. Semant. Knowl. Grid*. IEEE, 2009, pp. 254–257.
[7] R. G. Gallager, “Low-Density Parity-Check Codes,” p. 112, 1963.
[8] V. Kumar and O. Milenkovic, “On graphical representations of algebraic codes suitable for iterative decoding,” *IEEE Commun. Lett.*, vol. 9, no. 8, pp. 729–731, 2005.
[9] S. Sankaranarayanan and B. Vasic, “Iterative Decoding of Linear Block Codes: A Parity-Check Orthogonalization Approach,” *IEEE Trans. Inf. Theory*, vol. 51, no. 9, pp. 3347–3353, Sep. 2005.
[10] J. Feldman, M. Wainwright, and D. Karger, “Using Linear Programming to Decode Binary Linear Codes,” *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 954–972, Mar. 2005.
[11] S. Boyd, “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers,” *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.
[12] S. Barman, X. Liu, S. Draper, and B. Recht, “Decomposition methods for large scale LP decoding,” *2011 49th Annu. Allert. Conf. Commun. Control. Comput.*, vol. 59, no. 12, pp. 253–260, 2011.
[13] X. Liu, S. C. Draper, and B. Recht, “Suppressing pseudocodewords by penalizing the objective of LP decoding,” in *2012 IEEE Inf. Theory Work.* IEEE, Sep. 2012, pp. 367–371.
[14] X. Zhang and P. H. Siegel, “Efficient iterative LP decoding of LDPC codes with alternating direction method of multipliers,” in *2013 IEEE Int. Symp. Inf. Theory*. IEEE, Jul. 2013, pp. 1501–1505.
[15] G. Zhang, R. Heusdens, and W. B. Kleijn, “Large Scale LP Decoding with Low Complexity,” *IEEE Commun. Lett.*, vol. 17, no. 11, pp. 2152–2155, Nov. 2013.
[16] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Interscience, 2005.