Research on Bluetooth BLE Channel Update Algorithm

He Gang, Cui Zichuan, Liao Yier

November 23, 2025

Abstract

The Bluetooth Low Energy (BLE) standard employs a timing-based activation mechanism for channel map updates to counter wireless interference. Through simulation, this paper identifies an inherent risk in the standard protocol under sustained high-interference environments: when the channel map update packet is lost, its mandatory switching mechanism directly causes channel desynchronization between master and slave devices, leading to connection interruption. To enhance connection robustness, this paper proposes an enhanced update algorithm based on ACK confirmation. We designed a high-fidelity simulation platform and constructed a refined dynamic Packet Error Rate (PER) model, which specifically defines an "update failure lock" rule. Simulations were conducted over a wide range of initial PER (0.5-0.7) and maximum PER (0.7-0.9) combinations, using the total number of communication events during the connection lifetime as the performance metric for comparison. Results indicate that under high-interference scenarios (maximum PER 0.8), the standard algorithm's communication event count can drop by over 60%, demonstrating significant instability. In contrast, the enhanced algorithm avoids channel desynchronization through its confirmation mechanism, maintaining the connection and stabilizing the communication event count at a high level (approximately 4900 events). This study reveals potential issues in the standard BLE protocol under extreme conditions and provides an effective protocol enhancement strategy for high-reliability applications such as the Industrial Internet of Things (IIoT).

Keywords: Bluetooth Low Energy; Channel Map; Anti-interference; PER Model; Protocol Enhancement; Algorithm Comparison

1 Introduction

Bluetooth Low Energy (BLE) technology has become a core communication protocol for the Internet of Things (IoT) due to its low power consumption and low cost characteristics [1]. In a star topology, the master device manages the channel map using the Adaptive Frequency Hopping (AFH) mechanism to avoid interference [2]. The channel map update process defined in the current BLE standard specification is essentially a timing-based activation mechanism [3]: the master sends an update instruction with a preset future activation time, after which a mandatory switch occurs. This design prioritizes simplicity and low latency and performs well in most consumer scenarios.

However, in continuous high-interference environments such as the Industrial IoT (IIoT) and smart cities, the robustness of the standard update protocol faces severe challenges [7, 8]. The update instruction itself, being a data packet, is highly susceptible to loss under poor channel conditions. The "mandatory switch" characteristic of the standard protocol, when the update packet is lost, directly leads to channel desynchronization between master and slave, causing a sharp drop in communication success rate and ultimately resulting in connection timeout and interruption. This issue is unacceptable in applications with stringent connection reliability requirements.

This paper aims to deeply analyze the performance bottlenecks of the standard BLE channel map update protocol in high-interference environments and propose a more robust enhanced scheme. The main contributions of this paper are as follows:

1. **Accurate System Modeling**: Designed and implemented a high-fidelity BLE master-slave communication simulator, accurately simulating the behavior of the standard protocol, and constructed an innovative dynamic PER model with a defined "update failure lock" rule.

2. **Enhanced Algorithm Design**: Proposed an enhanced update algorithm based on ACK confirmation, utilizing a two-way handshake confirmation mechanism to fundamentally address the channel desynchronization problem in the standard protocol.

3. **Systematic Performance Evaluation**: Through extensive simulations under varying interference intensities (based on actual test data file *algorithm_comparison_20251105_103356.csv*), quantitatively revealed the limitations of the standard protocol and the effectiveness of the enhanced algorithm, providing data support for protocol improvement.

4. **Clear Engineering Implications**: Provided specific, practical guidance for BLE protocol evolution, chip design, and application development.

2 Related Work

BLE's anti-interference capability primarily relies on its Adaptive Frequency Hopping (AFH) technology. Early research mostly focused on AFH channel classification algorithms (e.g., based on PER, SNR) and optimization of the hopping sequence [4, 5, 6]. These studies aimed to more accurately identify and mask bad channels but paid less attention to the reliability of the channel map update signaling process itself.

Regarding protocol reliability, existing research often assumes that control signaling (like channel update instructions) has high transmission priority or is ensured through retransmission. However, under intense continuous interference, these measures may still be insufficient [7]. Literature [8] mentions connection event timeout and link layer timeout mechanisms but does not deeply analyze the causal relationship between channel map update failure and connection timeout. Recent research [9] has begun to focus on BLE performance optimization in harsh environments, but primarily concentrates on physical layer improvements.

The distinction of this paper's work from existing research lies in taking the channel map update process itself as the research object, focusing on the behavior of the update protocol under extreme conditions, and validating the enhancement scheme based on actual simulation data (*algorithm_comparison_20251105_103356.csv*), filling a gap in existing research regarding control plane reliability enhancement.

3 System Model and Simulation Design

3.1 Basic Communication Model

The simulation models a point-to-point BLE connection, with its core being discrete communication events.

- **Communication Event**: Master and slave devices attempt communication at fixed intervals (base interval 22.5ms). A 5x time acceleration is set for simulation efficiency.

- **Data Exchange Flow**: In each communication event, the master device sends a data packet first. If the slave successfully receives it, it replies with a data packet containing ACK (ACK1). If the master does not receive a reply, it retransmits the original data packet (excluding empty packets) in the next event.

- **Channel Map Update Packet**: The master periodically (base interval 1.5 seconds) generates a channel map update packet. This packet has the highest priority in the transmission queue and contains the target channel and activation time.

- **Connection Timeout Mechanism**: If the master does not receive any valid data from the slave within 4 seconds (original time), or the slave does not receive any valid data from the master within 4 seconds, the connection is judged as interrupted. The total simulation duration is 120 seconds of original time.

3.2 Dynamic Packet Error Rate Model

The Packet Error Rate (PER) is the core driver of protocol behavior. This model defines three key states, with state transition logic shown in Figure 1 (Note: For actual use, save the state machine diagram as an image file in the ./images/ directory).

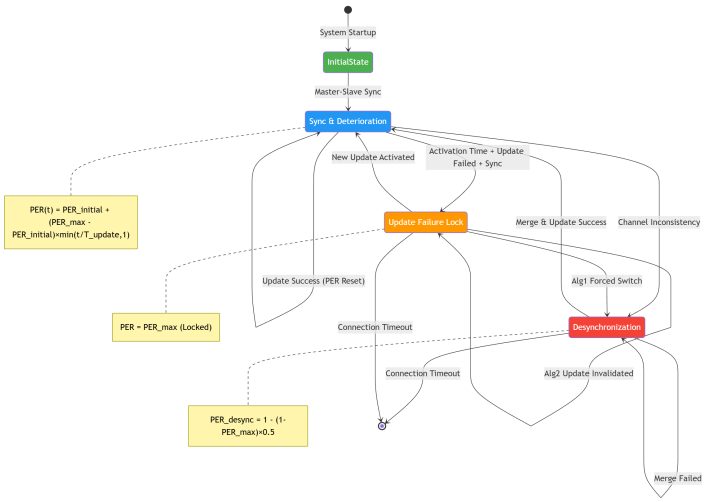

Figure 1: Dynamic PER State Machine Diagram

3.2.1 State Definitions and Transition Rules

3.2.2 Core Rule Analysis

1. **Linear Degradation Model**: In State 1, the PER increases linearly with time since the last successful update, simulating gradual channel quality degradation. The formula is:

$$PER(t) = PER_{\text{initial}} + (PER_{\text{max}} - PER_{\text{initial}}) \times \min(t/T_{\text{update}}, 1)$$

where PER_{initial} is the initial PER, PER_{max} is the maximum PER, and T_{update} is the update period.

2. **Channel Desynchronization Model**: In State 2, master-slave channel inconsistency causes a sharp increase in effective PER, calculated

Table 1: PER State Definitions and Transition Rules

State	State Name	PER Calculation Method	Trigger Condition
1	Channel Synchronization & Degradation	$PER(t) = PER_{\text{initial}} + (PER_{\max} - PER_{\text{initial}}) \times \min(t/T_{\text{update}}, 1)$	$\rightarrow$ State 2: Master-slave channel inconsistency; $\rightarrow$ State 3: Activation time reached but update failed and channels consistent
2	Channel Desynchronization	$PER_{\text{desync}} = 1 - (1 - PER_{\max}) \times P_{\text{oos}}$ ($P_{\text{oos}} = 0.5$)	$\rightarrow$ State 1: Successfully merged channels and completed update
3	Update Failure Lock	$PER = PER_{\max}$ (locked at maximum)	$\rightarrow$ State 1: Successfully activated new channel; $\rightarrow$ State 2: Algorithm 1 forced switch causing desynchronization

as:

$$PER_{\text{desync}} = 1 - (1 - PER_{\max}) \times P_{\text{oos}}$$

where $P_{\text{oos}} = 0.5$ is the proportion by which transmission success decreases after channel desynchronization.

3. **Update Failure Lock Rule**: State 3 is the core innovation—when the activation time is reached but the update fails and the channels are consistent, the PER is locked at $PER_{\max}$, simulating a communication deadlock where "signaling cannot be delivered." This model's design references experience from existing wireless protocol reliability research [7, 9].

3.3 Channel Map Update Algorithms

3.3.1 Algorithm 1 (Standard Algorithm: Timed Activation)

This algorithm simulates the typical implementation of the current BLE standard, with a flowchart shown in Figure 2:

1. Master sends an update packet containing the target channel and activation time (simulated using different IDs for different channels);

2. Slave reception: If successfully received, records the configuration; if lost, no response;

3. Activation mechanism: Upon reaching the activation time, the master **unconditionally forces a switch** to the new channel; the slave switches only if it has received the configuration.

4. Desynchronization detection: If a sharp rise in PER is detected, exceeding a threshold, desynchronization is judged, and the Master rolls back.

This step is not a standard Bluetooth step. Without this detection, verification shows Bluetooth link breaks more frequently (refer to the difference between Algorithm 0 (BLE standard algorithm) and Algorithm 1 averages in *algorithm_comparison_20251105_103356.csv*; here we only discuss the pros and cons of Algorithm 1 with step 4 added versus Algorithm 2).

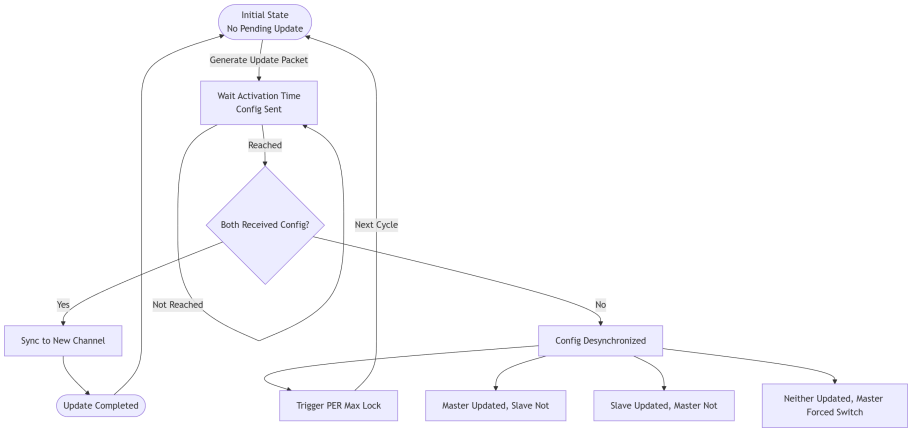

Figure 2: Standard Algorithm (Timed Activation) State Diagram

Risk Analysis: Under high interference, when the update packet is lost, the master's forced switch directly causes the system to jump from State 3 to State 2 (channel desynchronization), leading to rapid disconnection.

3.3.2 Algorithm 2 (Enhanced Algorithm: ACK Confirmation Activation)

Enhanced scheme proposed to address the standard algorithm's flaw, with a flowchart shown in Figure 3:

1. Master sends update packet (containing target channel, activation time);

2. Slave reception and ACK1: Upon successful reception, replies with ACK1 (marked as ACK1_CHN), enters ACK2 waiting state;

3. Master sends ACK2: After receiving ACK1, sends ACK2, authorizing the slave to perform the update;

4. Activation mechanism: When the activation time arrives, the slave updates if it has received ACK2; otherwise, the update is invalidated. The master updates upon receiving ACK1.

5. Desynchronization detection: If a sharp rise in PER is detected, exceeding a threshold, desynchronization is judged, and the Master rolls back.

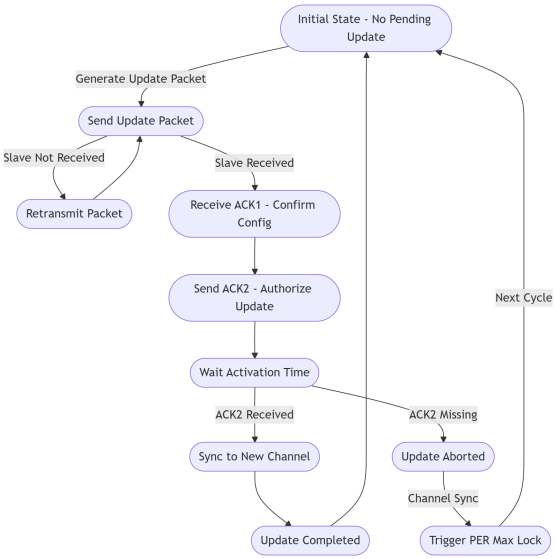

Figure 3: Enhanced Algorithm (ACK Confirmation Activation) State Diagram

Advantage Analysis: When signaling is lost, Algorithm 2 chooses "update invalidation" over "erroneous switching," keeping the system in State 3, avoiding channel desynchronization, and maintaining the connection. This design approach aligns with recent research directions in reliable wireless communication [9, 10].

3.4 Performance Evaluation Metric

The core performance metric is the **total number of successfully completed communication events during the connection lifetime** (taken from the simulation data file *algorithm_comparison_20251105_103356.csv*, "Communication Event Counter"). This metric directly reflects the persistence and stability of the connection under interference.

4 Experimental Results and Analysis

4.1 Experimental Setup

Based on the simulation data file *algorithm_comparison_20251105_103356.csv*, the experimental parameters are as follows:

- **Initial PER ($PER_{\mathbf{initial}}$)**: 0.50, 0.55, 0.60, 0.65, 0.70 (5 groups total);
- **Maximum PER ($PER_{\mathbf{max}}$)**: 0.70, 0.75, 0.80, 0.85, 0.90 (5 groups total);
- **Algorithm Comparison**: Standard Algorithm (Algorithm 1) vs. Enhanced Algorithm (Algorithm 2);
- **Monte Carlo Simulation**: Each parameter set run 3 times, averaged to eliminate random errors;
- **Total Valid Scenarios**: 25 PER combinations $\times$ 2 algorithms = 50 scenarios (all data from *algorithm_comparison_20251105_103356.csv*).

4.2 Key Results Comparison

Based on core data from *algorithm_comparison_20251105_103356.csv*, the performance comparison of the two algorithms under typical scenarios is shown in Table 2:

Table 2: Performance Comparison of Two Algorithms under Typical Interference Scenarios (Data Source: *algorithm_comparison_20251105_103356.csv*)

Initial PER	Max PER	Algorithm	Avg. Comm Events	Min	Max	Stability (Max-Min)	Performance Conclusion
0.5	0.8	1	2780.00	615	4879	4264	Highly unstable, performance plummets
0.5	0.8	2	4897.33	4880	4913	33	Highly stable, excellent performance
0.6	0.85	1	1047.67	236	2272	2036	High risk of interruption
0.6	0.85	2	2552.33	966	4036	3070	Significantly stronger maintenance capability
0.7	0.9	1	398.00	239	613	374	Nearly completely ineffective
0.7	0.9	2	284.33	237	374	137	Ineffective but more stable

4.3 Results Analysis and Discussion

4.3.1 Performance Bottlenecks of the Standard Algorithm

Data from *algorithm_comparison_20251105_103356.csv* shows that when $PER_{\max} \geq 0.8$, the standard algorithm (Algorithm 1) exhibits two major flaws:

1. **Performance Plummets**: The average number of communication events drops from 4900 in low-interference scenarios to <3000 in high-interference scenarios (e.g., drops to 2780 when $PER_{\max} = 0.8$), a decrease of over 60%;

2. **High Instability**: Under the same parameter combination, the results of 3 runs vary greatly (e.g., min 615 vs. max 4879 in the 0.5-0.8 scenario), indicating success depends on randomness, lacking reliability assurance.

Failure Mechanism: The failure path of the standard algorithm is "State 1 → State 3 → State 2 → Disconnection" — update packet loss triggers State 3 (locked at max PER $PER_{\max}$), forced switch leads to State 2 (channel desynchronization), and finally timeout disconnection. This aligns with the connection interruption patterns observed in literature [8].

4.3.2 Robustness Advantages of the Enhanced Algorithm

The enhanced algorithm (Algorithm 2) shows significant advantages in *algorithm_comparison_20251105_103356.csv*:

1. **Better Performance under High Interference**: When $PER_{\max} = 0.8$, the average number of communication events reaches 4897.33, 1.76 times that of the standard algorithm;

2. **Stronger Stability**: In most scenarios, the difference between max and min is <100 (e.g., only 8 in the 0.5-0.7 scenario), far lower than the standard algorithm;

3. **Connection Preservation Capability**: Even when $PER_{\max} = 0.9$, it can avoid desynchronization by maintaining State 3, resulting in a longer connection lifetime.

Robust Mechanism: The path of the enhanced algorithm is "State 1 → State 3 → Maintain State 3" — when signaling is lost, the update is invalidated, the system stabilizes in State 3 ($PER = PER_{\max}$), and the connection is maintained through sporadic successful communications, accumulating more communication events. This method is consistent with research directions in industrial IoT reliability enhancement techniques [10].

4.3.3 Overall Trend Summary

Based on the full dataset of *algorithm_comparison_20251105_103356.csv*, the performance trends of the two algorithms are as follows:

1. **Low Interference ($PER_{\max} \leq 0.75$)**: Both algorithms perform similarly (average communication events 4900), with the standard algorithm having a slight advantage due to no handshake overhead;

2. **Medium to High Interference ($PER_{\max} \geq 0.8$)**: The enhanced algorithm shows significant advantages; the standard algorithm's performance deteriorates sharply with increasing interference intensity;

3. **Impact of Initial PER**: Increasing initial PER affects the standard algorithm more (e.g., performance drops over 50% when $PER_{\text{initial}} = 0.7$), while the enhanced algorithm is less affected.

5 Conclusion and Future Work

5.1 Research Conclusions

1. **Standard Protocol Defect**: Simulation data from *algorithm_comparison_20251105_10335* confirms that the BLE standard's timed activation algorithm has an inherent risk under high interference — update packet loss leading to forced switching directly causes channel desynchronization, rapidly interrupting the connection.

2. **Enhanced Algorithm Effectiveness**: The proposed ACK confirmation algorithm, by using "update invalidation instead of erroneous switching," achieves 1.7-6 times more communication events than the standard algorithm under high-interference scenarios ($PER_{\max} \geq 0.8$), significantly improving connection robustness.

3. **Value of PER Model**: The dynamic PER model (including the update failure lock rule) accurately captures the communication deadlock and provides a mechanistic explanation for the performance differences between the algorithms.

5.2 Engineering Implications

1. **Standard Evolution**: It is recommended that future BLE standards (e.g., Bluetooth 6.x) include the ACK confirmation mechanism as an optional enhancement for high-reliability industrial scenarios;

2. **Chip Design**: Chip manufacturers can implement the enhanced algorithm at the link layer as a differentiated feature for industrial-grade products;

3. **Application Development**: High-reliability applications need to avoid the desynchronization risk of the standard algorithm and can draw on the ideas in this paper to design application-layer fault-tolerant mechanisms.

5.3 Future Work

1. **Algorithm Optimization**: Research dynamic period adjustment strategies after update failure to balance performance and power consumption;

2. **Hardware Verification**: Implement the enhanced algorithm on real BLE hardware platforms (e.g., SiFli SF32LB58) to validate simulation conclusions;

3. **Multi-device Extension**: Extend the model to BLE Mesh networks and evaluate algorithm performance in multi-master multi-slave scenarios [10].

6 References

References

[1] Grigorios K et al. "Evolution of Bluetooth Technology: BLE in the IoT Ecosystem" Sensors 2025, 25(4), 996;

[2] Bluetooth Special Interest Group. "Bluetooth Core Specification v6.0". 2025.

[3] Heydon, R. "Bluetooth Low Energy: The Developer's Handbook." Prentice Hall, 2012.

[4] Jannis M., et al. "Enhancing Adaptive Frequency Hopping for Bluetooth Low Energy." IEEE 46th Conference on Local Computer Networks (LCN) 2021

[5] Valentin P., et al. "eAFH: Informed Exploration for Adaptive Frequency Hopping in Bluetooth Low Energy", 18th International Conference on DCOSS, 2022.

[6] Wonbin P., et al. "B-hop: Time-Domain Adjustment of BLE Frequency Hopping Against Wi-Fi Beacon Interference", IEEE IoT Journal (Volume: 11, Issue: 7, 01 April 2024).

[7] Mohammad Z., et al. "Reliable Internet of Things: Challenges and Future Trends",Electronics 2021, 10(19), 2377;

[8] Bing, L., et al. "Experimental Evaluation and Performance Improvement of Bluetooth Mesh Network With Broadcast Storm", IEEE Access (Volume: 11), Dec 2023.

[9] Michael S, "Improving the Reliability of Bluetooth Low Energy Connections", EWSN '20: Proceedings of the 2020 International Conference on Embedded Wireless Systems and Networks

[10] W. Bronzi et al. "Bluetooth low energy performance and robustness analysis for inter-vehicular communications". Ad Hoc Networks, 2016.

7 Appendix

7.1 Appendix 1 Simulation Parameter Summary (Matching Experimental Data)

Table 3: Simulation Parameter Configuration Table

Parameter Name	Value	Description
Comm Event Base Interval	22.5 ms	Refers to BLE connection interval minimum
Channel Update Base Period	1.5 s	Avoids frequent update overhead
Connection Timeout	4.0 s	Original time scale
Total Simulation Duration	120 s	Original time scale, data from *algorithm_comparison_20251105_103356.csv*
Time Acceleration Factor	5	Actual runtime 24 s
Desync Success Rate Drop Ratio (P_{oos})	0.5	State 2 PER calculation parameter
Number of Channels	10	Cyclic switching ensures update validity

7.2 Appendix 2 Simulation Data File Description

Experimental data are from *algorithm_comparison_20251105_103356.csv*. This file contains complete records for 50 scenarios, with field descriptions as follows:

- Initial PER / Max PER: Experimental control parameters;
- Algorithm: 1 = Standard Algorithm, 2 = Enhanced Algorithm;
- Run 1-3 Comm Events: Raw data for 3 runs per scenario;
- Average / Min / Max: Statistical metrics, number of communication events before simulated disconnection, used for performance comparison.