Article

ACE: A Routing Algorithm Based on Autonomous Channel Scheduling for Bluetooth Mesh Network

Minyue Wang [1], Yeming Li [1], Jiamei Lv [1], Yi Gao [1], Cheng Qiao [2], Baiqiang Liu [2] and Wei Dong [1,*]

1 Zhejiang Provincial Key Laboratory of Service Robot, College of Computer Science, Zhejiang University, Hangzhou 310007, China; miny@zju.edu.cn (M.W.); liymemnets@zju.edu.cn (Y.L.); lvjm@zju.edu.cn (J.L.); gaoyi@zju.edu.cn (Y.G.)
2 Simon Electric (China) Co., Ltd., Shanghai 201103, China; cheng.qiao@simon.com.cn (C.Q.); brian.liu@simon.com.cn (B.L.)
* Correspondence: dongw@zju.edu.cn

Abstract: The Internet of Things (IoT) interconnects massive cyber-physical devices (CPD) to provide various applications, such as smart home and smart building. Bluetooth Mesh is an emerging networking technology, which can be used to organize a massive network with Bluetooth Low Energy (BLE) devices. Managed-flooding protocol is used in Bluetooth Mesh to route the data packets. Although it is a highly desirable option when data transmission is urgent, it is inefficient in a larger and denser mesh network due to the collisions of broadcast data packets. In this paper, we introduce ACE: a Routing Algorithm based on Autonomous Channel Scheduling for Bluetooth Mesh Network. ACE relies on the existing Bluetooth Mesh messages to distribute routes without additional traffic overhead and conducts a beacon-aware routing update adaptively as the topology evolves. In ACE, BLE channel resources can be efficiently utilized by a channel scheduling scheme for each node locally and autonomously without any neighborly negotiation. We implement ACE on the nRF52840 from Nordic Semiconductor and evaluate its effectiveness on our testbed. Compared to the Bluetooth Mesh, our experiments proved that ACE could reduce the end-to-end latency by 16%, alleviate packets collisions issues, and increase the packet delivery ratio (PDR) by 30% under heavy traffic. Moreover, simulation results verified that ACE has better scalability when the size and density of networks become larger and denser.

Keywords: Bluetooth Mesh; wireless mesh network; Internet of Things; BLE; routing protocol

check for updates

Citation: Wang, M.; Li, Y.; Lv, J.; Gao, Y.; Qiao, C.; Liu, B.; Dong, W. ACE: A Routing Algorithm Based on Autonomous Channel Scheduling for Bluetooth Mesh Network. *Electronics* **2022**, *11*, 113. https://doi.org/10.3390/electronics11010113

Academic Editors: Tony Jan, Deepak Puthal, Anca Ralescu and Mukesh Prasad

Received: 20 November 2021
Accepted: 27 December 2021
Published: 30 December 2021

1. Introduction

The past several years have witnessed the rapid development of Internet of Things (IoT) and edge computing [1] technologies. The IoT interconnects the cyber-physical devices (CPD) to provide various of services and applications, such as smart home and smart building. Bluetooth Low Power (BLE) is a widely used short-range low-power communication technology among the others, such as Zigbee, Z-Wave, Wi-Fi, and Thread [2]. BLE is originally designed to create a star-topology network, which severely limits the BLE network coverage. To overcome the limitation, significant efforts have been devoted by industry [3], academia [3]. Finally, the Bluetooth Mesh standard has been proposed by Bluetooth Special Interest Group (SIG).

In 2017, the Bluetooth Mesh version 1.0.1 specification [4] was released based on the existing lower layers of BLE by making use of its advertising capabilities. The Bluetooth Special Interest Group (SIG) has only considered a flooding-based protocol called managed flooding. Compared to other wireless communication protocols (including ZigBee, Thread, Z-Wave, Wi-Fi) that involve routing techniques, Bluetooth Mesh is more vulnerable to packet collisions as traffic intensity increase [5]. The advantages of managed flooding are its simplicity, redundancy, and flexibility. However, the broadcast storm [6] problem may occur in which the same message can be forwarded by different relay nodes multiple times.

Our design goal is to improve the performance of Bluetooth Mesh network, especially its inefficient managed flooding mechanism when delivering messages, and thus enable it to be applied to wider application scenarios with larger and denser network requirements like lighting control in a big building. In such a large-scale static network scenario, introducing a routing protocol can significantly improve network performances such as reliability and scalability. Still, no previous works have done this for standard Bluetooth Mesh to the best of our knowledge. However, compared to flooding, adding a routing algorithm without delicate design can bring some new problems, such as more management traffic overhead, higher latency, and less flexibility [3]. To this point, we summarize four main challenges as follows:

First, how to reduce the complexity of the routing algorithm and customize it for Bluetooth Mesh with minimal traffic overhead is a challenging issue. Running a routing algorithm is more expensive than flooding [3], There are computational and memory overhead to setup and maintain routing table. Bluetooth Mesh with complex protocol stack has the advantage of simplicity benefited from its managed flooding technique, thus adding an existing routing protocol to Bluetooth Mesh such as RPL [7] used in ZigBee is not a good choice. RPL is an oriented distance-vector routing protocol that organizes nodes in a Destination-Oriented DAG (DODAG) structure with large control overheads (e.g., DIO, DIS, DAO, and DAO-ACK). Suppose run RPL over Bluetooth Mesh, and nodes need to switch between connected and connectionless states frequently for control and data messages respectively [8] which significantly increase the network processing burden and seriously reduce network performances.

Second, it is challenging to perform a low end-to-end latency of routing algorithm for Bluetooth Mesh. Due to the potential multipath characteristic of flooding, an optimal path can be easily selected in standard Bluetooth Mesh, resulting in low end-to-end latency. Previous works focusing on routing algorithms for Bluetooth Mesh sacrificed the advantage of flooding and have a high end-to-end latency generally more than a second [8–10]. Therefore, they are unsuitable for delay-sensitive scenarios due to their time-consuming routing processes caused by frequent connection state transitions.

The third challenge is how to achieve an efficient BLE channel resource utilization in the routing algorithm for Bluetooth Mesh. Bluetooth Mesh utilizes only three ADV channels (37, 38, 39) for managed flooding, which increases the congestion risk of ADV channels. However, the rest 37 data channels are idle and wasted. This challenge can be abstracted as a CAP (Channel assignment Problem) [11], which is objected to maximize the channel resource utilization and minimize the interference. However, CAP is an NP-C problem, meaning that giving the optimal solution with an acceptable overhead is difficult. The data channels of BLE are used by two paired BLE devices through maintaining a frequency-hopping table after creating the BLE connection and then perform frequency-hopping operations on 37 data channels using the channel selection algorithm. Even though some previous routing algorithms for unofficial BLE mesh have taken 37 data channels into consideration, they were connection-oriented methods, which would bring an unacceptable delay by frequent connection establishments. Introducing TSCH (Time Slotted Channel Hopping) [7] to efficiently utilize channel resources by frequency-hopping operation needs a network-wide time synchronization at the beginning, which can cause an entire network collapse due to time synchronization errors, and a large delay can be brought by waiting for time slots when delivering packets.

Fourth, how to enable the routing algorithm more flexible and adapt to dynamic changes in mesh networks is challenging. Bluetooth Mesh can be easily applied to a dynamic network for its flexibility of managed-flooding. Conversely, topology changes can seriously affect routing networks' performance because frequent routes updating will interrupt communications and degrade network performance. Ad-hoc On-Demand Distance Vector Routing (AODV) [12] is a routing algorithm widely used in wireless mesh networks. It utilizes HELLO packets and RERR packets to cope with topology changes, which requires a relatively large traffic overhead to restore routes. However, we aim to design a routing algorithm adapted to both static and dynamic networks with minimal traffic overhead for better availability.

In this paper, we propose ACE: a Routing Algorithm based on Autonomous Channel Scheduling for Bluetooth Mesh Network. Four major advantages of ACE are shown as follows:

- Compared with Bluetooth Mesh, ACE can create and maintain routes without any additional route management packets nor connection state transitions.
- With the well-designed routing algorithm, the transmission latency of ACE is much lower than standard Bluetooth Mesh.
- The ACE can fully utilize all the 40 channels of BLE, but standard Bluetooth Mesh only use three advertising channels for data transmission.
- A beacon-aware routes recovery mechanism is proposed. It can recover routes mainly by adaptively listening beacon packets without any traffic overhead.

First, through reusing BLE Mesh messages (e.g., heartbeat message), Second, the routing algorithm can choose the route with minimum latency even in high dynamic scenarios. The end-to-end latency is greatly reduced compared with standard Bluetooth Mesh. Third, ACE integrated with autonomous channel scheduling scheme, which can utilize all the 37 data channels. Last, we create a beacon-aware routes recovery mechanism, which can enable movable nodes recovering routes mainly by listening beacon packets on a previously known channel adaptively without any traffic overhead.

We implement ACE on the nRF52840 SoC [13] and evaluate its performance both on our testbed and simulation, and also evaluate Bluetooth Mesh for comparison. The experimental results show that our routing algorithm on the nRF52 reduces the end-to-end latency by 16%, increases packet delivery ratio (PDR) by 30% under heavy traffic, improves scalability, BLE channel resource utilization, and can be less affected by external BLE interference.

The contributions of this work are threefold.

- We propose a proactive routing algorithm for Bluetooth Mesh network. And an autonomous channel scheduling mechanism is applied to fully utilize all the 40 BLE channels.
- We customize the routing algorithm for Bluetooth Mesh containing the following advantages: (1) Route creation reuses existing mesh messages without inner-node negotiation nor path reservation, and (2) routes recovery using an adaptively beacon-aware mechanism without additional traffic overhead.
- We implement ACE on the nRF52840 SoC, and the experimental results show that ACE outperforms Bluetooth Mesh in the following aspects: end-to-end latency, reliability, spectrum utilization, and scalability. Moreover, unmodified smartphones can control the mesh network without knowing the existence of ACE.

2. Background

This work proposes a novel routing algorithm for Bluetooth Mesh network. It is realized mainly by fully exploiting the features of Bluetooth Mesh Protocol Data Units (PDUs) and the BLE communication process. This section briefly introduces the basic background of Bluetooth Mesh to understand the rest of this paper better.

2.1. Overview

The Bluetooth Mesh specification [4] defines the mesh profile as a networking technology built on top of BLE from 2017, which introduces a new Bluetooth topology paradigm for many-to-many communications and allows to construct of large-scale sensor networks. Bluetooth Mesh uses a managed-flooding method to enable message delivery. To avoid the broadcast storm and control the message forwarding times, Bluetooth Mesh introduce TTL (Time To Live) field in mesh packets, and it should be decreased for each hop. The standard has several network features, including relay, proxy, friend, and low power. Two different bearers such as advertising bearer and GATT bearer for the non-connection-oriented bearer and connection-oriented bearer separately.

2.2. Bluetooth Mesh Stack

Bluetooth Mesh network stack consists of the following layers:

- **Bluetooth Low Energy Core Specification:** Bluetooth Mesh stack is built on top of the BLE specification.
- **Bearer Layer:** This layer defines exchange messages among nodes using ADV and GATT bearers, where ADV carries most of the traffic in the network.
- **Network Layer:** This layer defines addressing of transport messages, rules to relay, accept or reject messages, network message format, and encryption or authentication.
- **Lower Transport Layer:** This layer defines how to handle message segmentation and reassembly of transport PDUs.
- **Upper Transport Layer:** This layer defines how to handle encryption, decryption, and authentication of application data.
- **Access Layer:** This layer defines the format of application data, controls data encryption and decryption, maintains network context and application keys.
- **Foundation Model Layer:** This layer defines models to configure and manage a mesh network.
- **Model Layer:** This layer defines standardized application models such as lightning and sensor.

2.3. Managed Flooding

Bluetooth Mesh specification defines a message delivery method named managed flooding. It is based on the flooding algorithm, which is no need for any complex routing policy, thus provide a resilient network due to its high redundancy and multi-path capabilities. The following techniques is introduced to limit flooding communication:

- **Time-To-Live (TTL):** TTL restricts the number of hops a message can survive in the mesh network.
- **Message Caching:** This means mesh nodes can cache received packets to avoid unnecessary re-relaying of these packets.

2.4. Bluetooth Mesh Protocol Fields

Different layers of the Bluetooth Mesh stack define their own fields for Bluetooth Mesh packets. According to mesh profile specification [4], Figure 1 shows part of Bluetooth Mesh PDU structure at the different layers. The LE uncoded PHY contains a six-byte ADV address of the sending node. Inside the network PDU, there also involve address fields named SRC and DST address. These two fields represent the source element address and destination element address of each message and occupy two bytes, respectively. The TTL field is contained in the network PDU, which has a maximum value of 127.

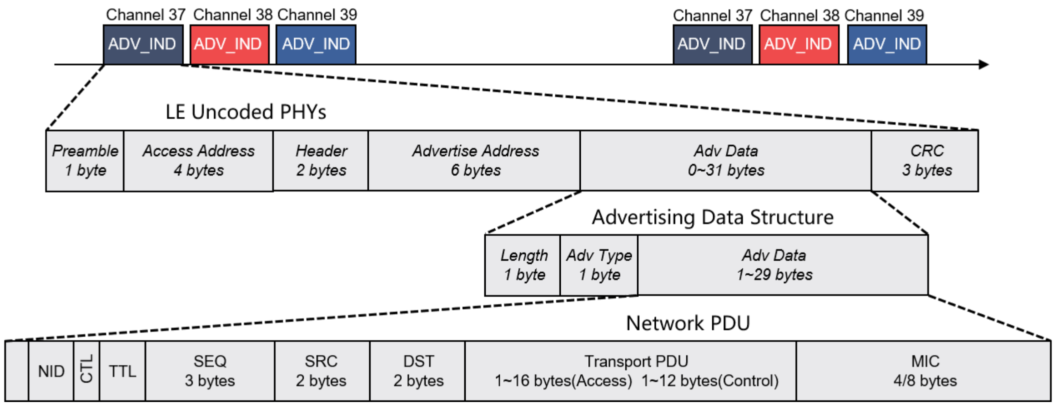


Figure 1. Bluetooth Mesh message encapsulation. The data packets are based on the BLE link-layer advertising data packet.

3. Bluetooth Mesh Communication and Problems

Bluetooth Mesh is designed to use a flooding-based communication technique. Thus, all messages transmissions in a Bluetooth Mesh network are broadcast under BLE's advertising and scanning scheme. BLE defined three channels (37, 38, 39) for all non-connected state communications and using non-connectable and non-scannable undirected advertising events.

During the advertising process, packets are broadcast sequentially over the three advertising frequency channels within the span called *Advertising Event*. All devices in radio range scanning on a matching ADV channel. The scanning process, called *Scanning Events* is conducted periodically. At the beginning of a *Scanning Event*, the scanning device starts listening on the next ADV channel from the last scanning channel. *ScanWindow* parameter defines the duration in which the link-layer scans on one ADV channel. *ScanInterval* defines a time interval between scans on different ADV channels. If the *ScanWindow* is equal to the *ScanInterval*, it means the device is continuously scanning. In Bluetooth Mesh, the time between the beginning of two consecutive advertising PDUs within an advertising event shall be less than or equal to 10 ms. The time between the start of two consecutive advertising events is controlled by the advInterval parameter (>=20 ms) with a random delay of no more than 10 ms.

Figure 2 depicts a communication flow between two adjacent relay nodes.

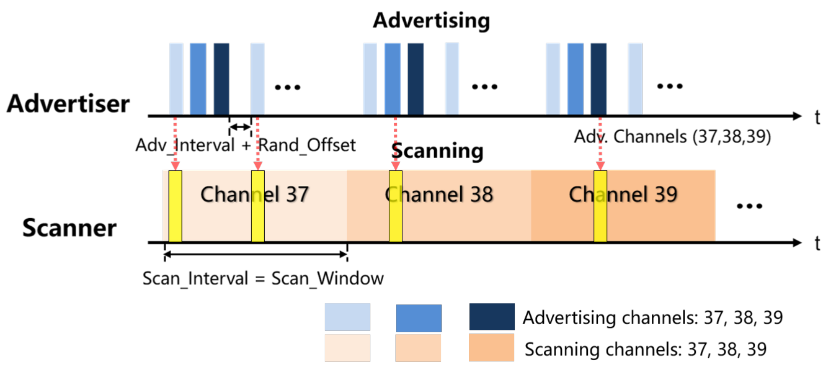

Advertising
Advertiser
...
t
Adv_Interval + Rand_Offset
Scanning
Adv. Channels (37,38,39)
Scanner
Channel 37
Channel 38
Channel 39
Scan_Interval = Scan_Window
Advertising channels: 37, 38, 39
Scanning channels: 37, 38, 39

Figure 2. Example of communication flow between two adjacent relay nodes in a Bluetooth Mesh network.

As introduced in the previous section, Bluetooth Mesh transmission model is based on *managed flooding*. In particular, nodes in the Bluetooth Mesh network can receive all messages within direct radio range, and messages will be relayed to reach the destination node in the distance. The Time-To-Live (TTL) field in the network PDU indicates the maximum number of hops. The design of managed flooding allows messages to reach their destination via multiple paths through the network, potentially improving the robustness performance. However, flooding is prone to suffer from collisions, especially when the network has a large number of packets. To alleviate this problem, Bluetooth Mesh makes it possible to contain random back-off periods before the start of an *Advertising Event* and randomize the time between the three transmissions within the *Advertising Event*.

We conclude several issues of Bluetooth Mesh through the analysis of the above communication process as follows:

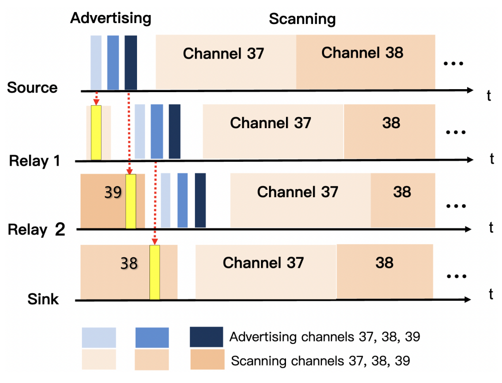
- **Inefficient BLE channels utilization:** Bluetooth Mesh uses only three of 40 channels of BLE channels for communication, which results in the congestion problem of three ADV channels, especially when data traffic becomes heavier. Meanwhile, the other 37 channels are completely unused, causing the extreme imbalance of BLE channels resource utilization.
- **Potential collision issue:** Managed flooding in Bluetooth Mesh does not contain any routing algorithm, resulting in a large number of duplicate packets in the Bluetooth Mesh network and wasting of wireless channel resources. All packets sending from

mesh nodes compete for three ADV channels, which indicates potential packets collisions and influences the reliability [5] and timeliness of the data transmission.

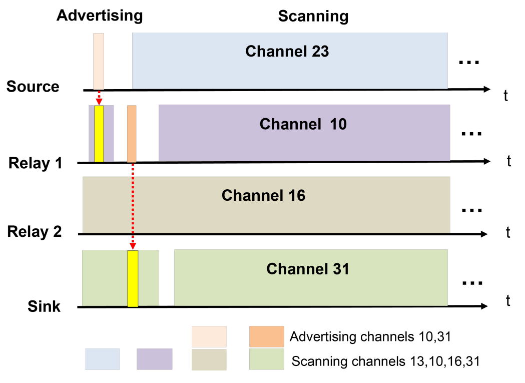
- **Noncontinuous scanning state:** According to Bluetooth Mesh network specification, the relays or the friend nodes should be scanning with a duty cycle as close as possible 100% to avoid missing any packets on advertising channels. However, when a scanner is switching scanning channels or processing received packets, it will leave the scanning state, and any packet received within these periods will be lost. In Particular, commuting from one scanning frequency to another takes up to 16 ms [14]. Thus, a noncontinuous scanning state will cause packet loss issue and reduce the Bluetooth Mesh network's reliability.
- **Uncertain end-to-end latency:** Communication occurs successfully in Bluetooth Mesh only if two nodes advertise and scan on the same channel at the same time, thus uncertain latency can be introduced by the time to match the ADV channel and the scanning channel. Besides, the path of packet transmission from the same source node to the same destination node may be different even in the same topology. Furthermore, packets collisions issue in Bluetooth Mesh could also cause an uncontrollable delay due to packets retransmission. The above problems are unacceptable for some applications with stable latency requirements.
- **Prone to be interference by BLE devices:** BLE devices are ubiquitous in our daily life such as smartphones, Bluetooth headsets, and fitness bands, etc. It is possible that various BLE devices from outside the Bluetooth Mesh network have a negative impact on the communication flows inside the network, especially from iBeacon devices using three ADV channels.

4. Motivation

To better understand our approach and show how the performance can be improved using our routing algorithm, we give an example topology and analyze the packet flows when delivering messages. Figure 3a,b show the case of Bluetooth Mesh and ACE, respectively.



(**a**) Bluetooth Mesh



(**b**) ACE

Figure 3. Examples of communication flows for Bluetooth Mesh and ACE. (**a**) The standard Bluetooth Mesh nodes advertise each data packet on three advertising channels. Each relay node will forward the data packets. (**b**) ACE nodes advertise the data packets only on the scanning channel of the next hop in the route.

At the left of Figure 3a,b, example topology and layout of the nodes are represented. The left source node is out of radio range from the right destination node. The relay node 1 and the relay node 2 in the middle are responsible for forwarding the message from the source node to the destination node. Green solid lines between two adjacent nodes represent the packet flows for the first hop while delivering a packet, and the red dashed lines represent the second hop accordingly. As shown in Figure 3a, relay 1 and relay 2 can both receive the packet from the source node and then forward the packet sequentially. For the second hop, as red dashed lines depicted, each node in the network can hear the packets from both relay 1 and relay 2. In particular, the source node receives the duplicate packets twice from two relay nodes, respectively, and two relay nodes could also receive the duplicate packet from each other. It's worth noting that, during the time of receiving duplicate packets, nodes would quit the scanning state and might lose packets that should have been received at that moment, which can reduce the reliability of Bluetooth Mesh network. In contrast, ACE has no duplicate packet to process, and thus there is no possibility of losing packets due to quitting the scanning state for duplicate packets receiving. ACE doesn't need to switch the scanning channels between three ADV channels, and thus improve the reliability due to leaving the scanning state. Besides, packet collisions issue in ACE is rarer than in Bluetooth Mesh.

A more detailed traffic flows are depicted in the right of Figure 3a,b for Bluetooth Mesh and ACE separately. In Figure 3a, the source node advertises packet on three advertisement channels at the beginning. Relay 1 receives the packet first because its scanning channel is 37, which is prior to the ADV channel 39 in an advertising event that relay 2 is scanning on. Next, relay 1 broadcast the packet on three ADV channels and so does relay 2 later. Finally, the destination node receives the packet on channel 38 from relay 1. As for ACE depicts in Figure 3b of the same scenario, after finishing setting up the routing table and selecting each corresponding scanning channel, relay 1, relay 2 and the destination node are scanning on the channel 10, 16 and 31, respectively. According to the routing and neighbor table, the source node forwards the packet to relay 1 by advertising the packet on channel 10. Meanwhile, relay 2 is unaware of this communication flow and continues to scan on channel 16. Similarly, relay 1 advertises the packet on channel 31, which the destination node is scanning on. There is neither duplicate packet received from unrelated nodes nor an uncertain delay introduced by overlapping the scanning and the advertisement channel during this process. The latency between the source node and the destination node with n relay are:

Bluetooth Mesh:

$$t = T_{send} + \sum_{i=1}^{n} (t_{scan,i} + t_{process,i} + t_{rand,i} + T_{send}) \tag{1}$$

ACE:

$$t = T_{send} + \sum_{i=1}^{n} (t_{process,i} + t_{rand,i} + T_{send}) \tag{2}$$

where T_{send} is the transmission time of an advertising packet, $t_{scan,i}$ is a scanning delay, depending on which ADV channel (37, 38, or 39) the packet is receiving on. $t_{rand,i}$ is ith relay node's random delay between receiving a mesh packet and relaying it, and $t_{process,i}$ is the packet processing time of ith relay node.

Compared to Equation (1), the delay formula in Equation (2) of ACE does not contain $t_{scan,i}$ item, which means the delay introduced by matching the scanning and the advertisement channel is reduced. Moreover, the probability of packets conflict in ACE is greatly reduced due to using more than ten times of scanning channels compared to three ADV channels using in Bluetooth Mesh.

5. ACE: A Routing Algorithm Based on Autonomous Channel Scheduling for Bluetooth Mesh Network

In the previous section, we showed that Bluetooth Mesh has five main problems and thus bring a negative impact on Bluetooth Mesh performance. To address these problems, our intuition is to introduce a routing algorithm for Bluetooth Mesh with design goals of low overhead, high reliability, low end-to-end latency, and also compatible with existing devices. This section presents the design and implementation aspects of our proposed routing algorithm.

5.1. Overview

Our routing algorithm follows these basic rules: (1) Reusing existing mesh packets (e.g., heartbeat packets) for route creation and maintenance, (2) scheduling BLE channel for each node locally and autonomously, and (3) recovering route for moved nodes by a beacon-aware scheme adaptively.

From a system integration perspective, we design ACE to be transparent to the applications. ACE is designed on the network layer and completely reuse Bluetooth Mesh stack layers above the network layer and the BLE specification core below it. Benefit from this, ACE enables off-the-shelf mobile phones to communicate with mesh network without knowing about the existence of ACE.

Figure 3a gives a communication flows example of Bluetooth Mesh from the source node to the destination node, and Figure 3b is the communication flows of ACE in the same scenario for comparison.

ACE can be divided into the following two phases:

Route creation Phase: In this phase, each node in Bluetooth Mesh creates a neighbor table and a routing table. A neighbor table records ADV addresses of all nodes within the radio range and their scanning channel, respectively. A routing table record the destination element address, the next hop's ADV address and the hop count.

Route Maintenance Phase: There are three actions involved in this phase:

- **Packet Sending:** When a node has a packet to send, it will first look up the routing table and find out the ADV address of the next hop, and then look up the neighbor table using the ADV address to find out the scanning channel of the next hop, finally send the packet on this specific channel. However, suppose the routing table does not have the entry of the destination node. In that case, the node will send the packet to all its neighbors to ensure the destination element can receive the packet at all possible.
- **Packet Receiving:** When a node has a packet to receive, the first thing to do is to check whether the destination element address is one of its element address. If so, it will deliver the packet to the upper layer for further processing directly. Otherwise, if the TTL value in the packet is more than one and its relay feature is on, the node will forward the packet and then start the packet sending phase as described above. Otherwise, the packet will be discarded for the TTL value is less than one.
- **Routing Update (optional):** As for typical Bluetooth Mesh applications like lighting control, all light nodes are static, and the network topology can always keep fixed. In such a widely used scenario, we can disable the routing update function to avoid unnecessary overhead. However, there are some dynamic Bluetooth Mesh network scenarios, such as asset tracking. The node mobility issue needs to be taken into consideration to keep the route valid all the time. Therefore, we design the routing update phase (optional) to enable our algorithm to adapt to a dynamic network as well. The details of the design are described in the next section.

5.2. Design and Implementation

We implement ACE on the nRF52840 SoC from Nordic Semiconductor. The design points of the four main modules are described below.

Route discovery: In the route creation phase, route discovery is done by sending heartbeat packets defined in the specification. Furthermore, every node setting the DST field of heartbeat packets to 0xFFFF, and the TTL value requires a tradeoff between the network size and node's memory capacity in terms of realistic condition. A larger initial TTL value means nodes can be discovered by the further nodes and thus increase the size of routing table. For example, if a Bluetooth Mesh network is large-scale, but the resource of nodes is constrained, we should set a small initial TTL value to limit the size of routing table. Especially if the TTL value in the received packets is exactly one less than the initial value, meaning the element address of SRC filed is one of its neighbors'. We record ADV address for each neighbor in the neighbor table after setting up a stable routing table.

Routing Table Generation: How to build a routing table for each relay node is a critical issue. Unlike most existing routing protocols, which need to record the entire routing path within a packet or send additional management packets to find the routing path. We explore the characteristics of BLE packets and novelly utilize two types of addresses in unmodified Bluetooth Mesh packets to simplify this process. One is the ADV address from PDU, and the other is the SRC address from the network PDU. The ADV address represents the address of next hop to the SRC element node, and the SRC element address represents the address of the destination element of this route entry. We choose minimum hop-count as the optimal routing metric for the latency reduction purpose. In the process of routing table creation, each node updates the ADV address of the next hop to the DST element by keeping the hop count minimal as a criterion. Concretely, each node obtains the hop count value by subtracting the TTL field in the received packet with a fixed initial TTL value. For example, if the received packet's TTL field is 125 and the initial TTL is set to 127, it means the packet has been relayed twice ($127 - 125 = 2$).

With the above two modules, we give the answer to the first challenge that reduces the complexity of the routing algorithm by reusing existing mesh messages without additional traffic overhead for route creation.

Channel Selection: How to deliver the packet to the next hop in terms of the routing table? Due to we do not consider a high delay and complex connection-oriented communication, sending messages to a specific next hop without overhearing by other nodes is a challenging issue. Novelly, We propose the following channel scheduling scheme, which operates autonomously to solve this problem: Each node should listen to one of the BLE channels after the network topology establishment phase. This specific channel is obtained by hashing the node's ADV address, which is expressed by:

$$Channel = \bmod\left(Hash(ADV\_addr(k)), N_{BLE\_channel} - 1\right) \tag{3}$$

where $N_{BLE\_channel}$ is the number of channels used by BLE and k represents a node in Bluetooth Mesh. $N_{BLE\_channel} - 1$ means reserving one channel for beaconing, which aims to cope with a dynamic network and will be explained further in the next part. The same hashing algorithm is used by all nodes in the network. Each node will use the algorithm to compute the scanning channels of all of its neighbors and record the computed results in the neighbor table. Now, if a node has a packet to deliver to the next hop, it should look up the neighbor table first and then send the packet to the channel, which the next hop node is scanning on.

By executing the above process, we can not only make full use of BLE channel resources but also alleviate the potential collision issue in Bluetooth Mesh. In addition, the end-to-end latency can be reduced in ACE owe to no channel matching delay introduced. To this point, we tackle the challenge two and three with our autonomous channel scheduling scheme.

Routing update: The routing algorithm described above can ensure a static network works very well. However, if a node in Bluetooth Mesh network moves, its neighbors have no idea about this change and continue sending packet on the specific channel to the moved node. Meanwhile, the moved node is keeping listening on its specific channel, but none of its new neighbors can send messages to it, causing the node to disconnect from the Bluetooth Mesh network. In order to adapt the routing algorithm to a dynamic network, we add a routing update module to guarantee the connectivity of the mesh network.

The main idea to deal with node mobility problem is to make the moved node aware of its movement adaptively and then rejoin the network by searching for beacon packets autonomously. A beacon packet reuses secure network beacon as defined in specification section 3.9.3 [4] to reduce the traffic overhead. Still, we modified it to be sent on a fixed channel such as 37. By doing this, the moved node can discover its new neighbors by scanning for beacon packets on the previously known beacon channel and then rejoin the mesh network. During this process, the communications between other nodes are unaffected, and the moved nodes can update its neighbor and routing table adaptively. Our design is briefly described below.

We set a routing update timer for a dynamic network. Each node should periodically send heartbeat packets to show its online state and periodically perform a routing update when the routing update timer timeout. It is worth noting that the TTL value in heartbeat packets sent in the route maintenance phase is set to 1, which means the node can be discovered only by its neighbor. In particular, each node should send a beacon packet periodically on a fixed beacon channel in case of the disconnection caused by mesh nodes' mobility. Concretely, we set an online timeout timer for each node. If a node does not receive any packet for several heartbeat packet periods, it means the node is most likely to be moved and disconnected from other nodes. So when the online timeout timer fired, the moved node will switch its scanning channel to the known beacon channel, then scans for beacon packets from neighbors. There are collisions between different beacon packets, especially in dense networkss. To avoid the continuous collision between multiple neighbors, we set some random backoff for the beacon packets retransmission. Besides, the interval of beacon can be tweaked according to the network density. When the network density is greater, the interval of beacon should be reduced to ensure the beacon can be received in a relatively short time. Similar to the route creation phase, the moved node update the neighbor table and then sends 'rediscovery packets' (heartbeat packets) whose TTL is set to the initial value in a shorter period. The purpose of doing so is to inform other nodes of this change as soon as possible. On the other hand, all nodes will update the routing table when the routing update timer timeout, which is very similar to the route creation phase. It is worth noting that the routing update timer period is much shorter than the duration of sending rediscovery packets, which can guarantee a solid and consistent route update of the entire network. Specifically, when the scale of the network is large, the setup and updating time of the routing table can be appropriately extended to ensure that the information of all nodes in the network is obtained and updated. Theoretically, the worst time of finishing updating for a new device will not exceed the sum of the expiration time of the routing table update timer and packets transmission time.

A brief process of routing update can be concluded as follows:

- First, the mobile node perceives the change of topological if it cannot receive the heartbeat message from the original neighbors on its scanning channel.
- The mobile node starts scanning the beacon message on specific beacon channel to discover the new neighbors.
- Once the discovery is done, the mobile node starts to transmit heartbeat message on the scanning channels of the new neighbors.
- The new neighbors will forward the heartbeat message to the entire network, and the other nodes will update routing table when their route update timer expires.

By means of the above mechanisms, the mesh network can maintain a connectable state adaptively without introducing additional traffic overhead. To this point, we give the solution to the first and fourth challenges.

Putting it All Together: We introduce three main algorithms of ACE. Algorithm 1 gives a high-level abstraction of the entire algorithm, containing two main phases, route creation phase, and route maintenance phase. Algorithm 2 invoked by Algorithm 1 and Algorithm 3 shows the core logic of routing table generation and update process. Algorithm 3 is invoked by Algorithm 1 and loops with the following three parts: (1) scheduling packets reception or transmission, (2) updating routes, and (3) detecting topology change and then recovering.

Combined with the above algorithms, we can conclude intuitively from the example depicted in Figure 3, ACE outperforms Bluetooth Mesh in end-to-end latency due to a direct channel mapping, and collisions issue can be alleviated for efficient traffic flows of ACE.

Algorithm 1: ACE

```
start_timer(initialize_timer);
publish heartbeat packet;
while initialize_timer is not fired do
    listen(ADV_CHANNELS);
    if pkt ← packet_RX() is not NULL then
        routing_table_pointer ← update_routing_table(pkt);
    end
    update_neighbor_table(routing_table_pointer);
end
listen(compute_channel(self.adv_addr));
routing_maintenance();
```

Algorithm 2: update_routing_table

```
Input: packet
Output: routing_table_pointer
dst_addr, adv_addr ← get_addr(packet);
hop ← compute_hop(packet, INIT_TTL);
for entry in routing_table do
    if dst_addr == entry.dst then
        if entry.hop_count > hop then
            update_routing_entry(dst_addr, adv_addr, hop);
        end
        break;
    end
end
if no dst_addr in routing_table then
    add_routing_entry(dst_addr, adv_addr, hop);
end
```

5.3. Discussion of Design Details

The previous section gives the general design and implementation of ACE, but there are still some design details that need to be discussed separately to present some performance tradeoffs:

Tradeoff between constrained resources and routing scale: One of the major characteristics of IoT devices is its constrained resources. Take nRF52840 SoC as an example. It has 256 KB RAM and 1 MB Flash, whose storage size is relatively large among similar chips. But it still can not afford a large routing table. In terms of ACE, each routing table entry include a 6-byte MAC address of next hop, 1-byte TTL, 2-byte element address of the next hop, and 4-byte pointer to the next table item. The RAM occupied by this data structure is 16 B since the memory is not aligned for ARM-GCC and SEGGER ARM compiler. Theoretically, the maximum size of the routing table is 16,000. However, we need to reserve some RAM for BLE protocol stack, application code, etc. The size of the routing table available in the real scenario is much smaller than 16,000 entries. Besides, the RAM of other mainstream BLE chip is much smaller than nRF52840. For example, 64 KB for nRF52832 and 88 KB for CC2642. The computational overhead increases as the number of table entries increase. We evaluate the searching time for routing tables with different sizes. The result shows that the searching time and routing table size are linearly proportional. The nRF52840 chip costs about 1.64 us the check one routing table entry. For a massive network with 10,000 nodes, the time for searching the routing table would be 16.4 ms for each hop, but the delay of one-hop data transmission is 24.7 ms. So, the size of the routing table needs to be carefully designed.

Algorithm 3: routing_maintenance

```
start_timer(online_timer);
start_timer(routing_update_timer);
while 1 do
    pkt_rx ← packet_RX();
    pkt_tx ← packet_TX();
    if online_timer is not fired then
        if pkt_rx is not NULL then
            timer_reschedule(online_timer);
            if update_entries_timer is start then
                routing_table_pointer ← update_routing_table(pkt_rx);
                update_neighbor_table(routing_table_pointer);
            end
            RX_process(pkt_rx);
        end
        if pkt_tx is not NULL then
            TX_process(pkt_tx);
        end
    end
    else
        start_timer(route_recovery_timer);
        while route_recovery_timer is not fired do
            listen(beacon_channel);
            pkt_beacon ← packet_RX() ;
            routing_table_pointer ← update_routing_table(pkt_beacon);
        end
        update_neighbor_table(routing_table_pointer);
        publish heartbeat packet;
        listen(compute_channel(self.adv_addr));
    end
    if routing_update_timer is fired then
        start_timer(update_entries_timer);
    end
end
```

Our solution is to manually change the initial TTL value of heartbeat packets during the route creation phase. This means that nodes only create routes for nodes within a certain hop count radius because packets with a TTL value of 0 can be discarded. If a node has to transmit messages to an element address with a hop count larger than the initial TTL value, according to our algorithm, the message will be broadcast to all of its neighbors at the beginning until some relays have the route entry of the destination address. This is equivalent to adding a multipath characteristic to distant nodes adaptively, improving the reliability performance. Therefore users should set an appropriate initial TTL value in terms of actual scenario requirements and hardware conditions, leading to a good tradeoff between the resource overhead and the routing scale.

Tradeoff between communication overhead and reliability: Co-existing with other 2.4 GHz wireless technologies such as Wi-Fi, BLE is prone to be interference by these signals. To reduce the interference issue, a dedicated frequency hopping mechanism on 37 data channels between two paired Bluetooth devices has been proposed by Bluetooth specification. In our algorithm, we can't conduct a frequency-hopping mechanism because ACE uses connectionless communication, which has no time synchronization between nodes so that a frequency-hopping sequence cannot be maintained.

Scanning on a fixed channel will reduce the communication reliability of nodes when selecting a poor-performing channel. Therefore, adaptively selecting a good link quality channel for each node is always a challenging problem. The BLE specification [15] foresees two mechanisms to improve the link-layer reliability: adaptive frequency hopping with channel blacklisting and physical (PHY) mode adaptation. However, they can only be used for connection-oriented BLE communication, which is not suitable for our low overhead routing algorithm. Besides, using a mechanism like retransmission can make up for the temporary poor link quality to ensure network reliability. For this reason, we do not add additional link quality estimation strategies to keep a low communication or management overhead for timely communication to our first algorithm version. In future work, we may filter the bad link quality channel for a higher reliability communication through blacklisting bad channel adaptively.

Tradeoff between compatibility and performance: A standard Bluetooth Mesh node conducts broadcast and scanning on three ADV channels, which is incompatible with ACE for its specific broadcast and scanning channel mechanism.

To enable our routing algorithm to be compatible with standard Bluetooth Mesh nodes, we should add at least one of the three ADV channels to each node's scanning and broadcast channel lists to send or receive messages from standard Bluetooth Mesh nodes. However, doing so can improve compatibility but significantly reduce network performance because multi-channel listening and broadcasting will sacrifice the advantage of low delay and reduce reliability. Therefore, better compatibility with existing Bluetooth Mesh devices remains future work.

6. Evaluation

Implementation: The hardware used to implement the ACE nodes were Nordic nRF52840 modules. Nordic provides its own implementation of the Bluetooth Mesh standard [16], which we used to implement ACE and conduct contrast experiments.

Configuration: We let node 1 in the front of Figure 4 act as the source node, and node 13 in the back of Figure 4 act as the destination node. If not special specified, all nodes in the network implement the generic on/off model, and the source node sends on/off message 1000 times to the destination node at a rate of 2 pkts/s. We use as low as −12 dBm transmission power, simulating a longer communication distance condition and making performance gaps between the two networks more significant in our limited experimental space.

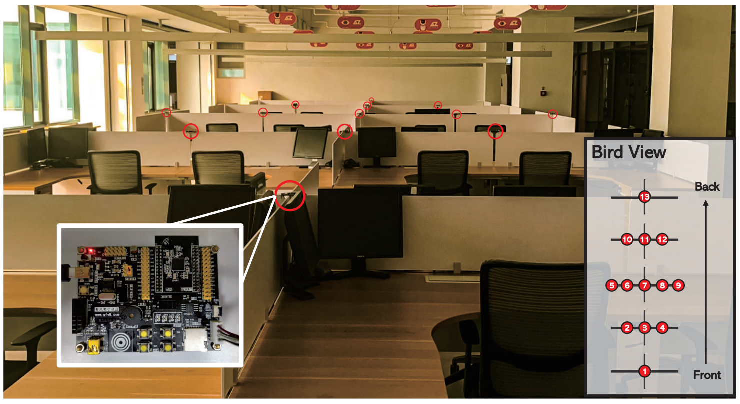


Figure 4. Testbed of 13 nodes in an empty office (18 m × 6 m).

Testbed: We deploy a testbed with 13 nodes of the type nRF52840 in a 18 × 6 = 108 m^2 empty office as depicted in Figure 4. The maximum distance between two nodes is close to 5 m, and the minimum distance is about 1.5 m. Although small, this setup represents both multi-hops and multi-neighbors scenarios, which can be generalized to a larger scene. neighbor **Metrics:** We focus on the following performance metrics:

- *Packet Delivery Ratio (PDR):* The PDR is the portion of packets sent at the application layer, which make it to their final destination, possibly over multiple hops.
- *End-to-end Latency:* The end-to-end latency is measured between the initial application's intention to send a packet and its reception at the final destination.
- *Route Setup Time:* The route setup time refers to the time when routing table entries are last updated by any of the nodes in the network.
- *Route Recovery Time:* The route recovery time refers to the time when the moved node receives the mesh packet sent to itself again.

6.1. Impact of Neighbor Numbers

In this section, we evaluate ACE performance by varying the number of neighbors. There are only five neighbor relay nodes (5, 6, 7, 8, 9) in use between the source node and the destination node in this scenario. We increase neighbor numbers from 1 to 5 and perform the same experiments that the source node sends on/off messages to the destination node for 1000 times. Figure 5 show the results.

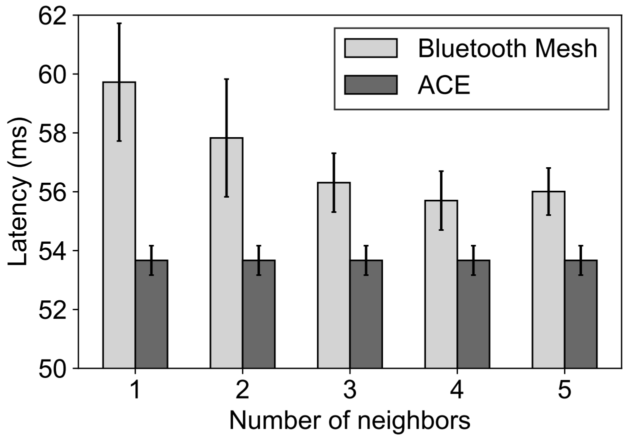


Figure 5. Comparison for the latency using a varying number of neighbors. ACE has an inherent low and stable latency characteristic due to the single-channel scanning and broadcasting scheme.

Figure 5 shows that ACE maintains a stable and low latency regardless of the number of neighbors in contrast to Bluetooth Mesh, which also has a larger variance of latency, verifying its performance improvement in end-to-end latency. In addition, the experiment results show that more neighbors can possibly reduce end-to-end latency. This is because more neighbors can lead to a higher probability of reducing the channel matching time among one of three ADV channels overall. However, the best latency condition in Bluetooth Mesh is just close to ACE because ACE has an inherent low and stable latency characteristic due to the single-channel scanning and broadcasting scheme.

6.2. Impact of Hop Numbers

Next, we investigate the impact of the hop count of packets before reaching the intended destination. It is more realistic for a mesh network to have communication flows that span multiple hops to arrive at the intended destination in larger deployment scenarios. To this end, we increase the number of relay nodes in which the packets can be forwarded by one to four relays and evaluate the Bluetooth Mesh and ACE performance, respectively. In particular, we only enable nodes 1, 3, 7, 11, 13 on the central axis to work to simplify a multi-hop scenario. The results are plotted in Figure 6. The performance gap between Bluetooth Mesh and ACE becomes more significant with the increase of the number of hops, which can be inferred from the lower and more stable latency of ACE with the rise of the number of hops compared to Bluetooth Mesh. In particular, ACE can reduce the latency by 16% and the delay variance by 80% in contrast to Bluetooth Mesh. These performance improvements benefit from the connectionless single-channel routing scheme of ACE.

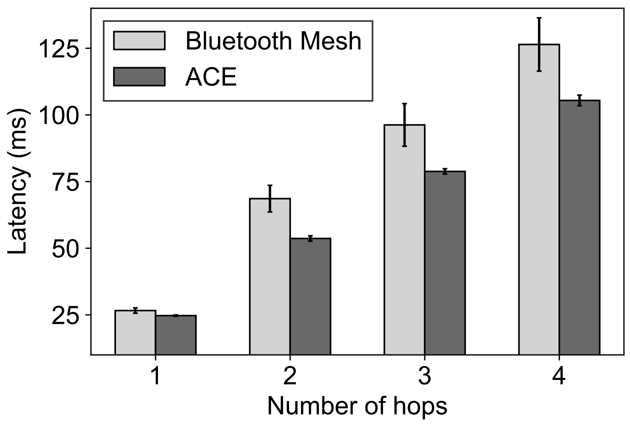


Figure 6. Comparison for the latency using a varying number of hops. The gap between the Bluetooth Mesh and ACE increased with the number of hops.

6.3. Impact of Traffic Load

Now we investigate the impact of traffic load. Figures 7 shows PDR metric when traffic load varies from 1 pkts/s to 10 pkts/s. The results in Figure 7 show that the PDR performance of both Bluetooth Mesh and ACE is rapidly degraded as traffic load increases from 1 pkts/s to 10 pkts/s. Bluetooth Mesh has a slightly higher PDR when network traffic is low due to its multi-path feature at the cost of packet redundancy. However, ACE maintains a relatively high PDR with a heavy traffic load. This is because the total amount of traffic in Bluetooth Mesh is heavier than ACE due to its managed-flooding scheme. Moreover, three ADV channels in Bluetooth Mesh are severely congested and prone to have packets collisions with such heavy traffic. In contrast, ACE using its channel-varying routing algorithm not only can reduce the number of packets in the whole network but also avoid the blind time caused by switching scanning channels or receiving duplicate packets. Overall, under a dense node deployment and a heavy traffic load (10 pkts/s), compared to Bluetooth Mesh, ACE achieves 30% higher PDR. Therefore, ACE can improve network reliability for applications with dense traffic.

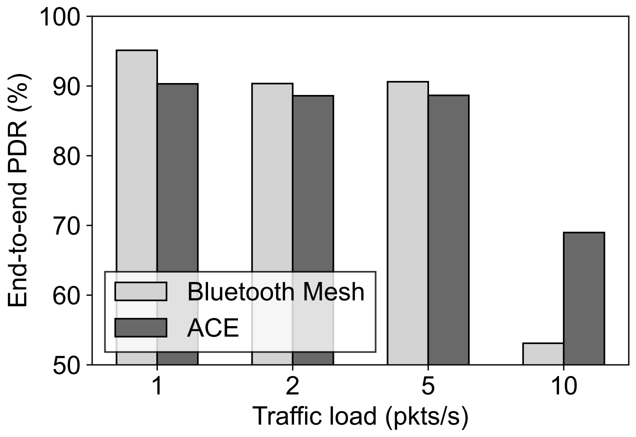
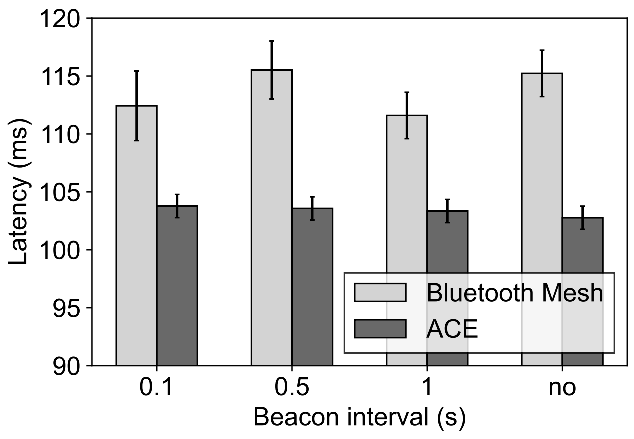


Figure 7. Comparison for the PDR using a varying of traffic load. The ACE has significant higher PDR in high traffic load scenario.

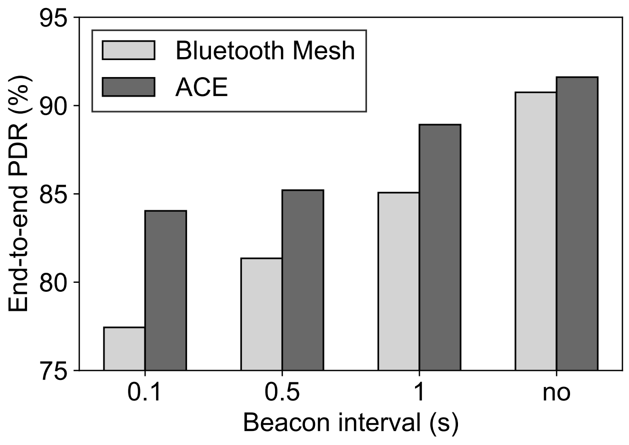
6.4. Impact of BLE Interference

The previous measurements were conducted in a controlled environment. To put it more realistically (e.g., in a smart-home environment), external BLE and other 2.4 GHz signals outside the Bluetooth Mesh network may affect the traffic inside the mesh network. Therefore, we investigate the impact of external BLE interferences by producing various intervals of BLE beacons. In this scenario, nodes 1, 3, 7, 11, 13 act as mesh nodes, and other nodes act as BLE beacon devices broadcasting packets on all three advertisement channels at different intervals, which simulate different levels of BLE interference.

Figure 8 summarizes the results. We notice that the PDR in Bluetooth Mesh drops with more BLE interference. Even though the PDR of ACE decreases as more interference is introduced, it generally performs better than Bluetooth Mesh. This is because, with more beacon interference, the probability of corruption due to packet collisions on three ADV channels increases, which has a negative impact on Bluetooth Mesh working only on three advertisement channels. Noting that the reason for the decrease of ACE's PDR with the same trend of Bluetooth Mesh is, node 7 happens to be scanning and advertising on advertisement channel 39 in our experiment, which can conflict with the beacon packets advertising on channel 39. However, this problem can easily avoid by selecting scanning channels among 37 data channels.



(**a**)



(**b**)

Figure 8. Comparison for the latency and PDR using a varying of external BLE interference. The performance of ACE is more stable since ACE can avoid most of the three advertising channels. (**a**) Latency. (**b**) PDR.

6.5. Impact of Network Size

In this section, we evaluate ACE performance with varying size of network. We simulated the impact of network size with different dense of nodes in a $20 \times 25 = 500$ m^2 grid topology shown in Figure 9. The source node sends on/off messages 1000 times to the destination node at rate of 10 pks/s.

The computational overhead will increase with the network size. we evaluate the time for nRF52840 to search routing tables with different size. The result is shown in Table 1. For the small scale network, the table searching time is negligible. However, in large scale networks, the routing table searching will have a greater impact on the transmission latency. Besides, for each hop of packet forwarding, all nodes need to search the routing table once, which greatly increase the multi-hop data transmission.

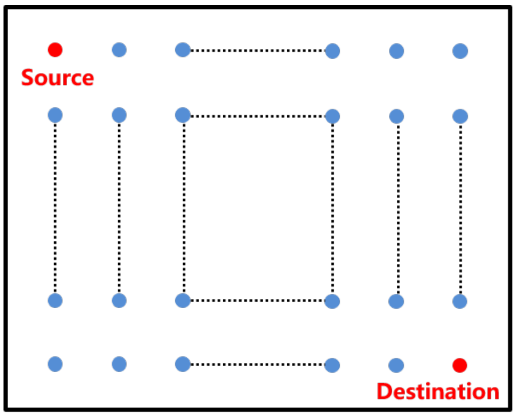


Figure 9. Grid topology used in simulation.

Table 1. Routing table searching time. The search time is linearly proportional to the size of the routing table.

Table Size	**10**	**100**	**500**	**1000**	**5000**	**10,000**
Searching Time (us)	18	165	822	1642	8205	16,408

The results in Figure 10 show that Bluetooth Mesh suffers a severe loss of reliability when the network size becomes larger and denser. In contrast, ACE maintains a high PDR despite the scale of the network, verifying better scalability of ACE than Bluetooth Mesh. The performance improvement in scalability benefits from efficient traffic flows of ACE, which can reduce the possibility of packets collisions and further improve network reliability.

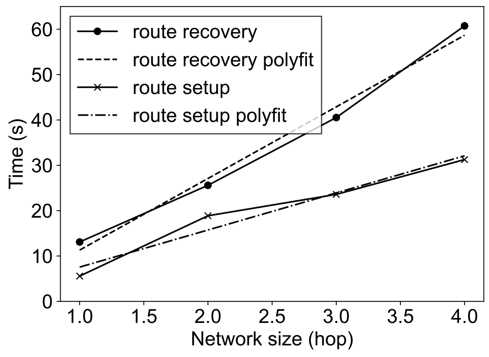


Figure 10. Performance of route setup time and route recovery time using a varying of network size. The recovery time of ACE is shorter than Bluetooth Mesh.

6.6. Route Setup Time and Route Recovery Time

In this section, we evaluate the ACE performance in a dynamic scenario with different network sizes. The network size is defined by varying amounts of hops, as shown in Figure 10. For example, nodes 1–9 form a three-hop network. We evaluate the topology setup time with different network sizes. In addition, We use a movable node and enable the source node to transmit packets to the movable constantly. We change the movable node position around the testbed and measure the time of receiving packets again since each movement, that is, the time of route recovery.

Figure 10 shows the results. Route setup can be done more quickly than the route recovery, it takes 31 s to set up the routing table for the entire testbed size (time of the last node to set up a stable route table), but the average time is only 14 s. As shown in Figure 10, the increasing route setup time is acceptable with the growth of the network size. As for route recovery time, it is increasing more significantly as the network size grows. The reason is that route recovery need to wait for each node's distributed routing update cycle; therefore, more nodes will lead to a longer time for the whole mesh network to finish updating routes process. We use linear regression to fit the route recovery time and route setup time. There is an obvious linear relationship between the scale of the network and the route setup and recovery time, and the variances are 3.64 and 3.83, respectively.

6.7. Compatibility with Unmodified Phones

Smartphones can access all models in standard Bluetooth Mesh network through the proxy nodes. As for ACE, since the stack layer and BLE core are not modified, the internal communication scheme is transparent for smartphones. We use the testbed to run ACE and control the generic on/off models with an unmodified smartphone. The result shows the unmodified smartphones are compatible with ACE network.

7. Discussion

Performance degradation in high dynamic scenario: Although ACE has low transmission latency and high PDR in the most common usage scenarios of Bluetooth Mesh (e.g., smart home, smart building). However, the topologies of these network are relatively static. The Figure 10 shows the ACE consume considerable time to achieve route recovery. As for the high-dynamic scenarios, such as the nodes are carried with humans or pets, the performance of ACE will decrease. To address this problem, the ACE can set up some proxy nodes which compatible with Bluetooth Mesh and ACE. The high-dynamic nodes can transmit data to the proxy nodes with Bluetooth Mesh protocol, and the proxy nodes forward the data packets into ACE network.

Various routing policies: For ACE, we choose the route which minimize the number of hops. Each node can utilize the TTL filed in the heartbeat message to infer the hops to other nodes autonomously, which is simple and effective. However, if other routing policy (e.g., minimize latency, minimize energy consumption) is applied, this method may fail. An intuitive solution is each node maintains a routing table with global information. But this will result in much control overhead during the route establishment and maintenance phase. Inspired by the THREAD protocol standard [17], ACE can adaptively choose a leader node which collects the global information of the network. The leader node should be placed where the network density is the highest to reduce the control overhead. The leader node can choose the optimal route or each node with a corresponding optimization model. For example, to minimize the transmission latency, the leader node can optimize the route by building a routing tree [18]. To optimize the energy consumption, the leader node should consider the energy consumption of channel hopping to assign optimal scanning channel for each node, then the energy-aware routing algorithm [19] can be used to generate the routes with minimum energy consumption. This improvement is reserved for future work.

Single link is susceptible to interference: In the ACE, we propose autonomous channel scheduling scheme to assign each node with a single scanning channel. However, without channel-hopping mechanism, the link between two nodes is vulnerable to the large amount of interference on 2.4 GHz ISM band. To solve the problem, we can assign three channels to each node, and the channel hopping mechanism can be applied with these three channels. The channel assign method can be varied. An intuitional way is to assign three continuous channels around the scanning channel assigned with autonomous channel scheduling scheme. Although the three channels are continuous, the link layer of BLE will mapped to three discontinuous physical channels. We reserve the channel-hopping based channel scheduling scheme as future work.

8. Related Works

Bluetooth Mesh Evaluation and Analysis: As a relatively new technology, most research works on Bluetooth Mesh have focused on the analysis or evaluation of its performance [2,5,14,20–23]. The experimental results showed that Bluetooth Mesh is vulnerable to external interference when using only 3 out of the 40 Bluetooth low-energy frequency channels [5]. However, the achievable average delay is relatively low [5,22], and it will be negatively impacted when the network is more sparse [22]. However, the scalability is especially challenging for Bluetooth Mesh since it is prone to suffer broadcast storm, hindering the communication reliability for denser deployments [5,20,22]. The optimization of Bluetooth Mesh network parameters such as retransmission parameters, buffer size, random backoff values, and TTL parameter [14] can be hot topics in the future. High energy consumption is a big issue for Bluetooth Mesh due to continuous scanning. An analytical model on important energy performance parameters of a battery-operated low power node is presented [21], and BMADS [24] is an asynchronous dynamic scanning mechanism designed to reduce the overall energy consumption of the mesh network.

Even though the above works have done various experiments and explored the disadvantages of Bluetooth Mesh, most of them do not propose improvement measures in terms of its deficiencies, such as collisions. ACE, in contrast, provides a novel routing algorithm for Bluetooth Mesh and improves its performance realistically.

Routing-based BLE mesh network: Before the Bluetooth Mesh standard was released in 2017, some research works have already been studied on a routing-based BLE mesh network. For example, a static tree topology over Bluetooth 4.0 is presented in [25,26], and Real Time BLE (RT-BLE) [26] is intended to enable bounded message delay for BLE mesh networks. MHTS [9] was designed over Bluetooth 4.0, based on next-hop, on-demand routing over the GATT layer. BLE Mesh Network (BMN) [10] uses the Directed Acyclic Graph (DAG) structure as a basis for routing, inspired by the IPv6 Routing Protocol for Low power and lossy networks (RPL). An on-demand routing for Bluetooth 4.1 was presented for forming scatternets in [27] . However, these BLE mesh solutions are not designed over the standard Bluetooth Mesh network and can introduce an unacceptable latency due to connection-oriented routing. ACE, in contrast, support routing algorithm for standard Bluetooth Mesh in a completely connectionless way, and it is more practical for its simplicity, timeliness, and scalability.

Ad Hoc Network Routing Protocols: Existing routing protocols for ad hoc networks can be classified either as proactive or reactive. Classic proactive protocols include DSDV [28], TBRPF [29], and WRP [30]. These protocols continuously evaluate the routes within the network so that the route is already known when a packet needs to be forwarded. Also, there are Artificial Intelligence based management algorithm for improving high-dynamic network performance [31]. In contrast, reactive or on-demand protocols invoke a route determination procedure by an on-demand route query. Examples of reactive protocols include AODV [12], DSR [32], and TORA [33]. The on-demand discovery of routes can result in much less traffic than proactive schemes; however, the route acquisition delay can be significant. We do not apply any of the above protocols directly to Bluetooth Mesh network. Instead, we explored the features of Bluetooth Mesh protocol and customized the routing protocol for it with the essence of the above classic ad hoc network routing protocols.

Synchronous Transmissions: Synchronous transmission is a recent direction in low-power wireless networks for its high reliability, low latency, and energy efficiency. By exploiting constructive interference and capture effect, Glossy [34] can flooding by precisely timing transmissions. Even in the presence of duplications, the receiver can successfully receive packets. BlueFlood [35] present a Bluetooth version of the concurrent transmission with low power and reliable flooding. However, synchronous transmissions always need one central entity to control and initiate the synchronous transmissions. Moreover, they require accurate time synchronization across the network and are often limited in terms of generality, such as being restricted to periodic traffic. In contrast, ACE transmits messages without complex time synchronization or waiting for any time slot, achieving simplify operations and reduce end-to-end latency.

9. Conclusions

In this paper, we have analyzed some drawbacks of the standard Bluetooth Mesh network caused by its managed flooding mechanism. Its disadvantages mainly include the following main aspects, inefficient spectrum resource utilization, potential collision issue, and prone to be interference by BLE devices. In order to address these issues, we design and implement ACE, which creates and maintains routes without additional traffic overhead by reusing existing mesh packets. Moreover, an autonomous channel scheduling mechanism and a beacon-aware route recovery scheme are proposed. To the best of our knowledge, this paper presents the first work to design and implement a routing algorithm for the standard Bluetooth Mesh network. Our study proves that ACE has the potential to support a variety of applications, not only for static sensor network applications with low-rate traffic but also a dynamic network or applications with heavy traffic. In contrast, the standard Bluetooth Mesh is more suitable for a sparse and small-scale network due to its potential broadcast storm problem. Our experimental evaluation verifies that ACE outperforms Bluetooth Mesh in several aspects: latency, reliability, spectrum utilization, and scalability.

Author Contributions: Conceptualization, M.W.; methodology, M.W., J.L.; software, M.W., Y.L.; validation, M.W., Y.L.; formal analysis, M.W., W.D., Y.G.; investigation, M.W., J.L.; resources, C.Q., B.L.; data curation, M.W., Y.L.; writing—original draft preparation, M.W.; writing—review and editing, Y.L., W.D., Y.G.; visualization, M.W.; supervision, W.D., Y.G.; project administration, M.W.; funding acquisition, W.D., Y.G., C.Q., B.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Key R&D Program of China under Grant No. 2019YFB1600700, the National Science Foundation of China under Grant No. 62072396, and Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. LR19F020001.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lakhan, A.; Li, J.; Groenli, T.M.; Sodhro, A.H.; Zardari, N.A.; Imran, A.S.; Thinnukool, O.; Khuwuthyakorn, P. Dynamic Application Partitioning and Task-Scheduling Secure Schemes for Biosensor Healthcare Workload in Mobile Edge Cloud. *Electronics* **2021**, *10*, 2797. [CrossRef]
2. Yin, J.; Yang, Z.; Cao, H.; Liu, T.; Zhou, Z.; Wu, C. A survey on Bluetooth 5.0 and mesh: New milestones of IoT. *ACM Trans. Sens. Netw. (TOSN)* **2019**, *15*, 1–29. [CrossRef]
3. Darroudi, S.M.; Gomez, C. Bluetooth low energy mesh networks: A survey. *Sensors* **2017**, *17*, 1467. [CrossRef] [PubMed]
4. Bluetooth, S. *Mesh Profile Specification: 1.0*; Bluetooth Special Interest Group: Kirkland, WA, USA, 2017.
5. Rondón, R.; Mahmood, A.; Grimaldi, S.; Gidlund, M. Understanding the Performance of Bluetooth Mesh: Reliability, Delay, and Scalability Analysis. *IEEE Internet Things J.* **2019**, *7*, 2089–2101. [CrossRef]
6. Tseng, Y.C.; Ni, S.Y.; Chen, Y.S.; Sheu, J.P. The broadcast storm problem in a mobile ad hoc network. *Wirel. Netw.* **2002**, *8*, 153–167. [CrossRef]
7. Kinney, P.; Brethour, V.; Bain, J.; Houghton, P.; Lampe, J.; Brethour, V.; Sahinoglu, Z.; Orlik, P.; Lakkis, I.; Hach, R.; et al. IEEE Standard for Part 15.4: Wireless MAC and PHY Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs): Amendment 1: Add Alternate PHY. 2007. Available online: https://standards.ieee.org/standard/802_15_4-2015.html (accessed on 26 December 2021).
8. Lee, T.; Lee, M.S.; Kim, H.S.; Bahk, S. A synergistic architecture for rpl over ble. In Proceedings of the 2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), London, UK, 27–30 June 2016; pp. 1–9.
9. Mikhaylov, K.; Tervonen, J. Multihop data transfer service for Bluetooth Low Energy. In Proceedings of the 2013 13th International Conference on ITS Telecommunications (ITST), Tampere, Finland, 5–7 November 2013; pp. 319–324.
10. Sirur, S.; Juturu, P.; Gupta, H.P.; Serikar, P.R.; Reddy, Y.K.; Barak, S.; Kim, B. A mesh network for mobile devices using bluetooth low energy. In Proceedings of the 2015 IEEE SENSORS, Busan, Korea, 1–4 November 2015; pp. 1–4.
11. Audhya, G.K.; Sinha, K.; Ghosh, S.C.; Sinha, B.P. A survey on the channel assignment problem in wireless networks. *Wirel. Commun. Mob. Comput.* **2011**, *11*, 583–609. [CrossRef]

12. Perkins, C.E.; Royer, E.M. Ad-hoc on-demand distance vector routing. In Proceedings of the WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, USA, 25–26 February 1999; pp. 90–100.
13. Semiconductor, N. nRF52840 Development Kit. 2017. Available online: https://www.nordicsemi.com/Products/Development-hardware/nRF52840-DK (accessed on 26 December 2021).
14. Pérez-Díaz-De-Cerio, D.; García-Lozano, M.; Bardají, A.V.; Valenzuela, J.L. Bluetooth Mesh Analysis, Issues, and Challenges. *IEEE Access* **2020**, *8*, 53784–53800.
15. Woolley, M. Bluetooth Core Specification v5. Bluetooth. 2019. Available online: https://www.bluetooth.com/specifications/specs/core-specification/ (accessed on 26 December 2021).
16. Semiconductor, N. nRF5 SDK for Mesh, Software Development Kit for Bluetooth Mesh. 2019. Available online: https://www.nordicsemi.com/Products/Development-software/nRF5-SDK-for-Mesh (accessed on 26 December 2021).
17. THREAD Protocol Standard. 2021. Available online: https://www.threadgroup.org/ (accessed on 26 December 2021).
18. McQuillan, J.; Richer, I.; Rosen, E. The new routing algorithm for the ARPANET. *IEEE Trans. Commun.* **1980**, *28*, 711–719. [CrossRef]
19. Amgoth, T.; Jana, P.K. Energy-aware routing algorithm for wireless sensor networks. *Comput. Electr. Eng.* **2015**, *41*, 357–367. [CrossRef]
20. De Leon, E.; Nabi, M. An Experimental Performance Evaluation of Bluetooth Mesh Technology for Monitoring Applications. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea, 25–28 May 2020; pp. 1–6.
21. Darroudi, S.M.; Caldera-Sànchez, R.; Gomez, C. Bluetooth mesh energy consumption: A model. *Sensors* **2019**, *19*, 1238. [CrossRef] [PubMed]
22. Baert, M.; Rossey, J.; Shahid, A.; Hoebeke, J. The Bluetooth mesh standard: An overview and experimental evaluation. *Sensors* **2018**, *18*, 2409. [CrossRef] [PubMed]
23. Almon, L.; Álvarez, F.; Kamp, L.; Hollick, M. The King is Dead Long Live the King! Towards Systematic Performance Evaluation of Heterogeneous Bluetooth Mesh Networks in Real World Environments. In Proceedings of the 2019 IEEE 44th Conference on Local Computer Networks (LCN), Osnabrueck, Germany, 14–17 October 2019; pp. 389–397.
24. Perez-Diaz-de Cerio, D.; Valenzuela, J.; Garcia-Lozano, M.; Hernández-Solana, Á.; Valdovinos, A. BMADS: BLE Mesh Asynchronous Dynamic Scanning. *IEEE Internet Things J.* **2020**, *8*, 2558–2573. [CrossRef]
25. Maharjan, B.K.; Witkowski, U.; Zandian, R. Tree network based on Bluetooth 4.0 for wireless sensor network applications. In Proceedings of the 2014 6th European Embedded Design in Education and Research Conference (EDERC), Milan, Italy, 11–12 September 2014; pp. 172–176.
26. Patti, G.; Leonardi, L.; Bello, L.L. A Bluetooth low energy real-time protocol for industrial wireless mesh networks. In Proceedings of the IECON 2016—42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, 23–26 October 2016; pp. 4627–4632.
27. Guo, Z.; Harris, I.G.; Tsaur, L.F.; Chen, X. An on-demand scatternet formation and multi-hop routing protocol for BLE-based wireless sensor networks. In Proceedings of the 2015 IEEE Wireless Communications and Networking Conference (WCNC), New Orleans, LA, USA, 9–12 March 2015; pp. 1590–1595.
28. Perkins, C.E.; Bhagwat, P. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *ACM SIGCOMM Comput. Commun. Rev.* **1994**, *24*, 234–244. [CrossRef]
29. Bellur, B.; Ogier, R.G. A reliable, efficient topology broadcast protocol for dynamic networks. In Proceedings of the IEEE INFOCOM'99. Conference on Computer Communications Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, The Future is Now (Cat. No. 99CH36320), New York, NY, USA, 21–25 March 1999; Volume 1, pp. 178–186.
30. Murthy, S.; Garcia-Luna-Aceves, J.J. An efficient routing protocol for wireless networks. *Mob. Netw. Appl.* **1996**, *1*, 183–197. [CrossRef]
31. Sodhro, A.H.; Sodhro, G.H.; Guizani, M.; Pirbhulal, S.; Boukerche, A. AI-enabled reliable channel modeling architecture for fog computing vehicular networks. *IEEE Wirel. Commun.* **2020**, *27*, 14–21. [CrossRef]
32. Johnson, D.B.; Maltz, D.A. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 153–181.
33. Park, V.D.; Corson, M.S. A highly adaptive distributed routing algorithm for mobile wireless networks. In Proceedings of the INFOCOM'97, Kobe, Japan, 7–11 April 1997; Volume 3, pp. 1405–1413.
34. Ferrari, F.; Zimmerling, M.; Thiele, L.; Saukh, O. Efficient network flooding and time synchronization with glossy. In Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks, Chicago, IL, USA, 12–14 April 2011; pp. 73–84.
35. Al Nahas, B.; Duquennoy, S.; Landsiedel, O. Concurrent Transmissions for Multi-Hop Bluetooth 5. In Proceedings of the IEEE International Conference on Embedded Wireless Systems and Networks (EWSN), Beijing, China, 25–27 February 2019; pp. 130–141.