國立臺北科技大學

資訊工程系碩士班

碩士學位論文

改善 FruityMesh 藍牙低功耗網狀網路傳輸效能的設計與實作

Design and Implementation of Performance Enhancements in FruityMesh Based Bluetooth Low Energy Mesh Networks

研究生：陳柏勳

指導教授：吳和庭博士

中華民國一百一十四年七月

國立臺北科技大學

資訊工程系碩士班

碩士學位論文

改善 FruityMesh 藍牙低功耗網狀網路傳輸效能的設計與實作

Design and Implementation of Performance Enhancements in FruityMesh Based Bluetooth Low Energy Mesh Networks

研究生：陳柏勳

指導教授：吳和庭博士

中華民國一百一十四年七月

「學位論文口試委員會審定書」掃描檔

審定書填寫方式以系所規定為準，但檢附在電子論文內的掃描檔須具備以下條件：

1. 含指導教授、口試委員及系所主管的完整簽名。
2. 口試委員人數正確，碩士口試委員至少 3 人、博士口試委員至少 5 人。
3. 若此頁有論文題目，題目應和書背、封面、書名頁、摘要頁的題目相符。
4. 此頁有無浮水印皆可。

審定書不用頁碼

摘要

關鍵詞：藍牙低功耗、物聯網、FruityMesh、BLE Mesh、網路拓撲、傳輸壅塞、重傳率、封包傳遞成功率

藍牙低功耗（Bluetooth Low Energy, BLE）有省電及低成本的特性，使得藍牙技術在物聯網（Internet of Things, IoT）中占據重要的角色。在物聯網的應用中，大量使用無線感測網路（Wireless Sensor Networks, WSN），會在環境中分布建立許多的節點，而節點不只有當作感知器測量環境的數據，常常還要當作中繼節點，負責轉傳發送端與目的端之間的封包。最終，將所有量測的數據匯集到 Sink 節點，以監控所有的節點數據，將數據儲存後，進行分析後並做出適當的處理，也可以透過分析數據預測環境的變化，並提前做出適當的處理。本論文針對 FruityMesh 網路建立流程進行改善，讓 Sink 節點在網路建立完成後成為整個網狀網路的根節點，並且確保 Sink 節點斷線後重新連線時，仍然是根節點的角色。如此一來，所有非 Sink 節點在傳送封包至 Sink 節點時，只需要往父親節點傳輸封包即可，有效的減少節點發送及轉發的次數。此外，本論文也透過調整 BLE 相關參數，緩解大量封包匯聚至根節點所造成傳輸壅塞的問題，進一步提升了整體 Mesh 網路的封包傳遞成功率（Packet Delivery Ratio, PDR）、降低了封包傳輸延遲及重傳率，進而改善網路效能。

Abstract

Keyword: Bluetooth Low Energy,Internet of Things,FruityMesh,BLE Mesh,Network Topology,Transmission Congestion,Retransmission Rate,Packet Delivery Ratio

Bluetooth Low Energy （BLE）is characterized by its low power consumption and cost-effectiveness, making it a key technology in the development of the Internet of Things （IoT）. In IoT applications, Wireless Sensor Networks （WSNs）are widely used, where numerous sensor nodes are distributed throughout the environment. These nodes not only collect environmental data but often function as relay nodes, responsible for forwarding packets between the source and destination. Ultimately, all collected data is aggregated at the Sink node, which serves as a central point for monitoring, storage, and analysis. This enables the system to make informed decisions and even predict environmental changes based on the analyzed data. This thesis proposes an enhancement to the FruityMesh network formation process, ensuring that the Sink node becomes the root node of the mesh network once the network is established. Furthermore, it guarantees that the Sink node retains its root node role even after disconnection and reconnection. With this configuration, non-Sink nodes can transmit packets toward the Sink node by simply forwarding them to their parent node, effectively reducing the number of transmissions and retransmissions required. Additionally, by tuning specific BLE parameters, the proposed method alleviates network congestion caused by high packet traffic toward the root node. The experimental results show that the proposed improvements increase the Packet Delivery Ratio （PDR）, reduce transmission delay and retransmission rates, and ultimately enhance overall network performance.

致謝

所有對於研究提供協助之人或機構，作者都可在誌謝中表達感謝之意。

目錄

圖目錄

表目錄

第一章　緒論

1.1　研究背景

科技的快速進步，讓人們的生活更加便利，物聯網（IoT）的應用已經與日常生活密不可分，包含了醫療及工業的應用，無所不在，[1] 根據日商環球訊息有限公司（GII）調查，物聯網（IoT）市場規模預計從 2024 年到 2029 年，將從 1.17 兆美元增加至 2.37 兆美元，年均複合成長率（CAGR）為 15.12%，如圖**??**所示。

圖 1.1 IoT 市場規模評估 [1]

2010 年 6 月藍牙技術聯盟（Bluetooth Special Interest Group）提出了低功耗藍牙（Bluetooth Low Energy, BLE），BLE 省電及低成本的特性，使得藍牙技術在物聯網（IoT）的應用種佔據了不可或缺的角色，例如: 目前市面上的無線設備包括藍牙耳機、藍牙鍵盤及藍牙滑鼠。物聯網（IoT）的應用中，大量使用無線感測網路（Wireless Sensor Networks, WSN），會在環境之中分布許多的節點，而節點不只有當作感知器測量環境的數據，常常還要當作中繼節點，轉傳發送端與目的端的封包，最終將所有量測的數據匯集到終端節點，以監控所有的節點數據，將數據儲存後，進行分析後並做出適當的處理，

也可以透過分析數據預測環境的變化，並提前做出適當的處理。

藍牙網狀網路（Bluetooth Mesh）架構的實現，讓 BLE 更具有可靠性（Reliability）及擴展性（Scalability），可以允許多個 BLE 相互連接並形成網狀結構，讓封包可以在多個裝置或節點之間進行傳輸，讓傳輸距離不會受到單一裝置的傳輸範圍限制，解決了節點之間裝置連接數量的限制以及傳輸距離不足的問題，在物聯網（IoT）的應用，例如: 智慧建築、智慧工業、智慧城市、智慧家庭.. 等等，BLE 都已經扮演重要也不可或缺的角色。

在物聯網（IoT）與藍牙網狀網路（Bluetooth Mesh）中，對整個系統架構做出適當的評估，在不影響裝置效能的情況下，設計多個藍牙裝置之間的分流機制，因為在整個系統中，流量可能會有所起伏，為了讓每個裝置可以有一樣的傳輸品質，且系統可以發揮最好的吞吐量。

1.2 研究動機與目的

根據文獻 [2] 中針對 Multi-Hop 的提出 DOT 及 DOST 排程設計，研究發現在原生 FruityMesh 上封包傳輸是採用廣播的方式，向相鄰的所有節點發送封包，這種設計雖然簡單，卻容易導致大量冗餘封包的產生，這會導致節點之間傳輸不必要的封包，最終導致整個 BLE Mesh 出現廣播風暴。

[2] 作者針對 FruityMesh 傳輸時產生的廣播風暴，提出了 DOT（Destination Oriented Transmission）排程的機制，將 BLE Mesh 導入樹狀結構，對整個系統進行分層管理，讓封包傳輸只會向正確的路徑上傳輸，避免了廣播風暴的產生，但是作者在研究過程中，也發現 BLE Multi-Hop 網狀網路節點會擁有 Master 以及 Slave 兩種角色，而在同一個時間下，節點同時會是 Master 和 Slave 的狀態，而這種狀態上的衝突會導致的封包傳輸產生遺失現象。

[2] 作者為了解決角色衝突的問題，在 DOT 的架構下增加啟用以及禁用的機制，提出 DOST（Destination Oriented Switch Transmission）的排程設計，利用 TDMA 讓裝置在同一個時間只會扮演 Master 或 Slave，解決在同一個時間點節點可能會因為扮演 Master 及 Slave 角色衝突而導致的封包傳輸遺失的問題；在研究 [2] 中作者在 FruityMesh 中導入 DOST 設計，改善了大部分的封包傳輸延遲以及封包重傳率，但仍存在些許效能瓶頸，

因為在 FruityMesh 中，所有節點的封包傳輸都是透過廣播的方式進行，這會導致在多個節點同時傳輸封包時，可能會產生封包碰撞的問題，導致封包傳輸延遲增加。

此外，文獻指出，若要在 DOST 架構下進一步提升效能，需確保拓樸建立時，Sink 節點為整體網路的根節點。然而原研究並未針對拓樸控制機制提出具體解法。若未能確保 Sink 位於樹根，當其斷線並重新加入網路時，可能無法恢復為原先的根節點角色，進而影響整體傳輸路徑的穩定性與效能。

在 [2][3] 中，針對了 single hop mesh 及 multi hop mesh 的網路拓樸進行探討，當使用合適的封包連接參數可以有效降低封包傳輸延遲及重傳率，但是數據實際上傳輸時，每個封包大小並不固定，[4] 中，探討了不同封包大小可以算出最適合的連接參數，在傳輸過程動態調整連接的 CE(Connection Event) 及 CI(Connection Interval)，達到更好的吞吐量。

本計畫將研究中在 DOT 的排程設計下，仍然會有些許因為封包遺失，而產生的封包重傳的問題，以及在藍牙網狀網路（Bluetooth Mesh）架構，執行 TDMA 的排程機制，結合調整連線數據，透過修改 Connection Interval 數值，確保傳輸品質及確保系統可以產生最大吞吐量，並修改 FruityMesh 拓樸方法，讓 BLE Mesh 建立的過程確保 Sink 點在整個樹狀結構的根節點，也確保 Sink 點如果斷線後重新連線後依然是整個 BLE Mesh 樹狀結構的根節點。

1.3 論文架構

第二章　相關技術與背景

2.1　藍牙低功耗技術

藍牙技術聯盟（Bluetooth Special Interest Group）在 2010 年 6 月發布了可以短距離數據交換和低功耗的藍芽低功耗技術（Bluetooth Low Energy, BLE）。而藍芽低功耗技術被發布後，就被物聯網（IoT）廣泛的應用，包括了家庭娛樂、醫療保健、運動健身、安防以及信標等領域。

藍牙低功耗技術（Bluetooth Low Energy, BLE）是一種功耗極低的技術，這一個技術讓裝置在大部分的時間都在休眠模式，只有在需要使用該裝置時，才會快速喚醒進行工作，這讓 BLE 裝置僅需要一顆鈕扣電池就可以運作數月甚至數年之久，這讓 BLE 生產成本更低，且保留了傳統藍芽（Classic Bluetooth）類似的通訊範圍，且一樣相容於手機、平板電腦等設備。

BLE 運作在 2.4G 的 ISM 頻段，利用 FDMA（Frequency Division Multiple Access），將 2402MHz 至 2480MHz 分成 40 個 Channel，又將這些 Channel 又分成兩種傳輸模式，廣播模式（Advertising Mode）及連線模式（Connection-Oriented Mode），其中廣播模式使用了 Channel 37、Channel 38、Channel 39，工作頻率分別是 2403MHz、2426MHz、2480MHz，剩餘的 37 個 Channel 為連接模式使用，如圖2.1所示。

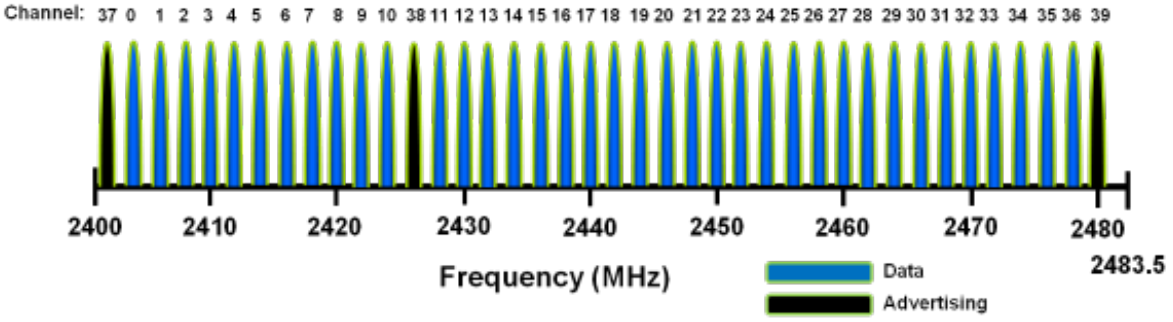

圖 2.1 廣播頻道與資料頻道示意圖 [5]

廣播模式和連接模式的運作機制由 BLE 的控制層（Controller Layer）狀態機管理，包括 Standby（等待）、Advertising（廣播）、Scanning（掃描）、Initiating（初始化）、Connection（連接）五種狀態，如圖2.2所示。

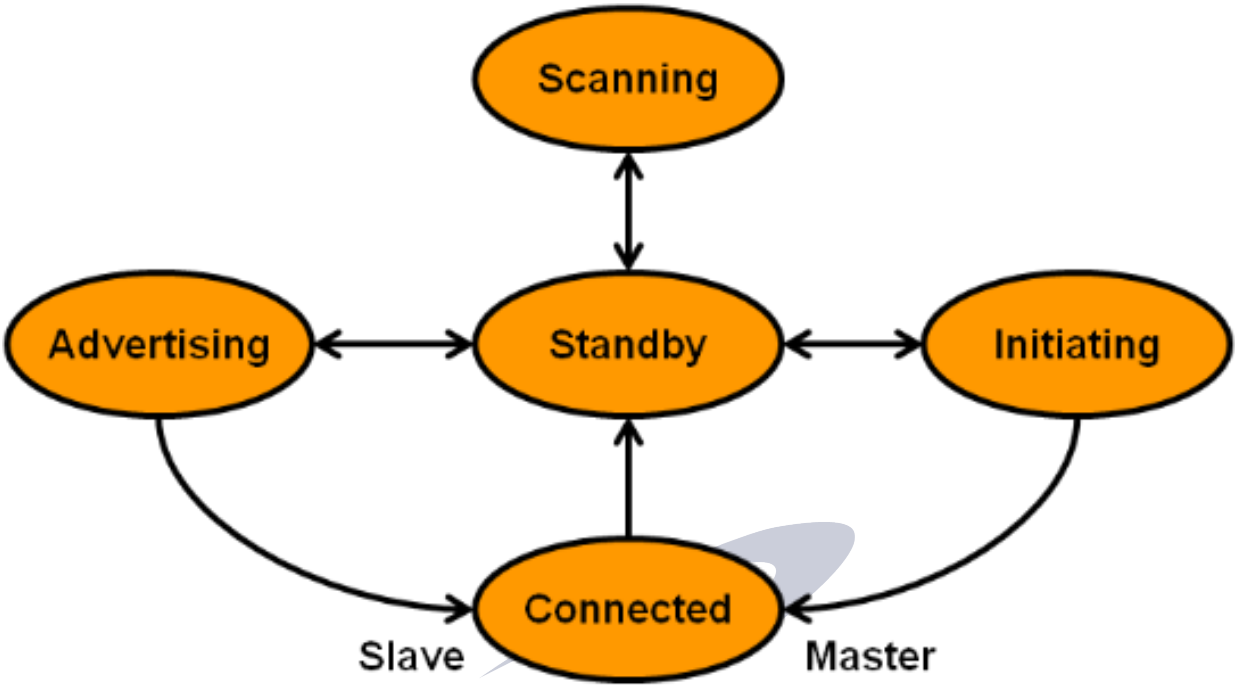

圖 2.2 Controller layer 狀態機示意圖 [5]

2.1.1 廣播模式

在廣播模式（Advertising Mode）中，會使用 Channel 37、Channel 38、channel 39 這三個廣播頻道，主要用於掃描裝置、建立通訊頻道和廣播的傳輸，其中廣播者（Advertiser）是由待機狀態（Standby Status）進入廣播狀態（Advertising Status），廣播者會在三個廣播頻道輪流發送廣播封包，讓掃描者（Scanner）可以檢測到其存在，並提供基本的數據，例如: 裝置名稱或狀態。

掃描者（Scanner）是由待機狀態（Standby Status）進入掃描狀態（Scanner Status），掃描者會輪流掃描三個廣播頻道的廣播封包，已接收範圍內的所有廣播的資訊，掃描收集數據後，準備與廣播設備建立連接，如圖2.3所示。

2.1.2 連接模式

連接模式（Connection-Oriented Mode）中，設備需要首先透過廣播模式建立連接，其中廣播者（Slave）負責發送連線請求的廣播封包，而掃描者（Master）檢測到廣播封包後，從 Standby 進入 Scanning，再進入 Initiating 狀態，開始與廣播設備握手（Handshake）。

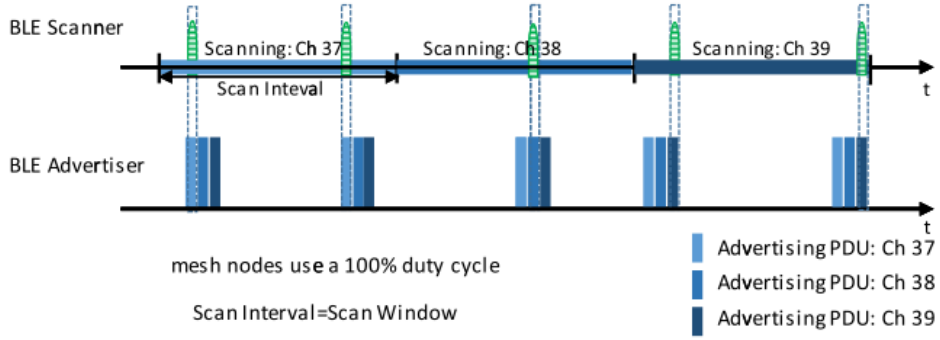

圖 2.3 廣播模式示意圖 [6]

當握手成功後，雙方進入 Connection 狀態。連接建立後，Master 與 Slave 可通過剩餘的 37 個數據頻道進行數據傳輸；Master 負責與多個 Slave 的連線管理，分配專屬的時間槽（Time Slot）給各連接設備，即使沒有數據需要傳輸，系統仍會保留固定的時間槽以確保系統的穩定性，避免通訊衝突。圖2.4為連線模式示意圖。

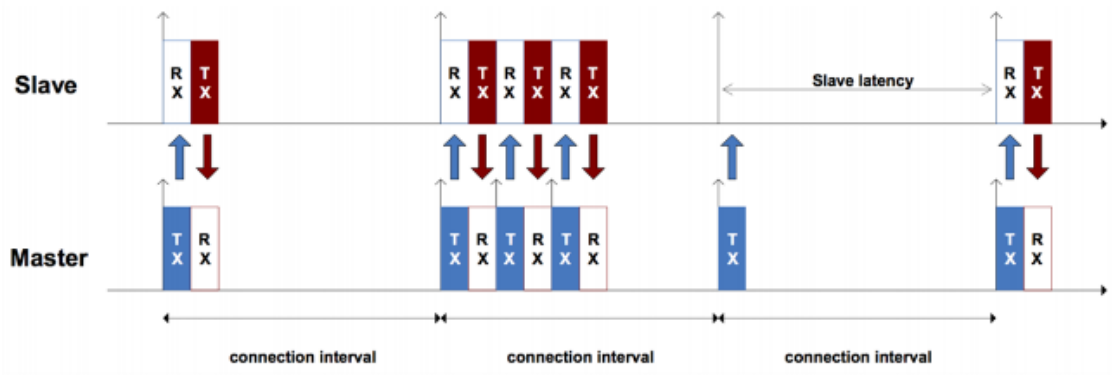

圖 2.4 連線模式示意圖 [6]

2.1.3 BLE 排程機制

廣播者（Advertiser）和掃描者（Scanner）會在建立連線後，進行數據的交換，當掃描者在三個掃描頻道掃瞄並偵測到廣播者發送的封包後，掃描者會在約 150 微秒（T_{IFS}）後回應一個 CONNECT_IND 封包。CONNECT_IND 封包包含了多項管理連接的參數，例如：影響錨點時間（Anchor Point, AP）的 WinSize 以及 WinOffset。

建立連線之後，Connection Interval （CI）、Anchor Point （AP）和 Connection Event（CE）對於裝置之間的排程機制影響非常的大，在一個 Connection Interval（CI）的時間內，一個 Slave 裝置只會與 Master 裝置有一個 Connection Event（CE）傳輸時間來進行資

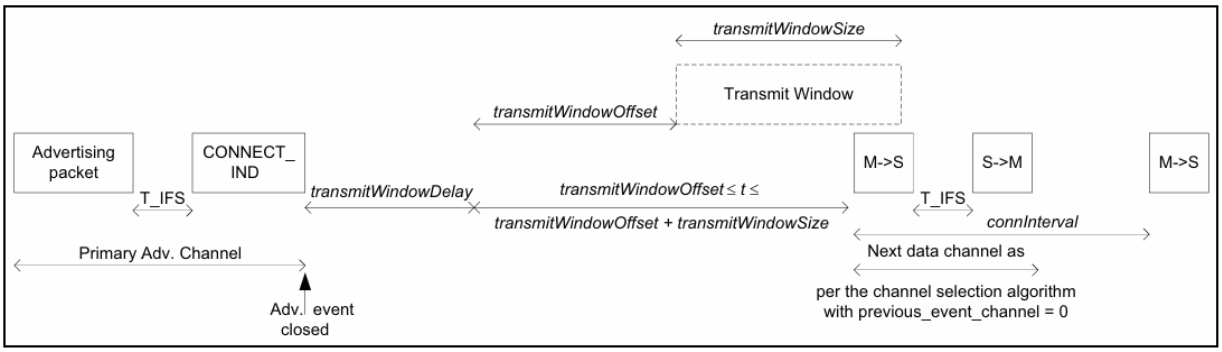

圖 2.5 Master 連接建立過程和連接事件 [7]

料的交換，所以 Connection Interval（CI）的時間影響 Connection Event（CE）的發生頻率，這直接影響到整個系統的效能及吞吐量，圖2.5表示了 Master 連接建立過程和連接事件。

CONNECT_IND 封包會在 t_{ind} 時間內完成傳輸，Master 會在公式2.1至公式2.2的時間內設置第一個錨點（Anchor Point, AP），其中 transmitWindowDelay 通常為 $1.25\,ms$，用來同步 Master 與 Slave。transmitWinOffset 與 transmitWinSize 分別由公式2.3與公式2.4算出。WinOffset 代表錨點（Anchor Point, AP）的偏移量；WinSize 代表錨點（Anchor Point, AP）可能發生的時間範圍 [8]。

$$t_{ind} + transmitWindowDelay + transmitWinOffset \tag{2.1}$$

$$t_{ind} + transmitWindowDelay + transmitWinOffset + transmitWinSize \tag{2.2}$$

$$transmitWinOffset = WinOffset \times 1.25\,ms \tag{2.3}$$

$$transmitWinSize = WinSize \times 1.25\,ms \tag{2.4}$$

2.2 藍牙網狀網路

藍牙網狀網路（Bluetooth Mesh）是由藍牙技術聯盟（Bluetooth Special Interest Group）於 2017 年發布的標準，旨在解決傳統藍牙技術在物聯網（IoT）應用中的限制。藍牙網狀網路允許多個 BLE 裝置之間建立一個可靠且可擴展的網狀結構，這使得裝置之間可以進行多跳傳輸，從而擴大了傳輸距離和覆蓋範圍。

Bluetooth Mesh 是基於藍牙低功耗（Bluetooth Low Energy, BLE）的網狀網路技術，是多個裝置進行多對多（many-to-many）連接的拓樸技術，可以將資料從 BLE Mesh 中的其中一個節點，發送至 BLE Mesh 中的任何一個節點，且任兩的節點間的傳輸路徑，並不會只有一條，因為這種多重路徑（multi-path）的特性，讓 Mesh 中如果有一個節點故障，也部會因此癱瘓整個系統。此技術有效解決了 BLE 在長距離和多設備連接上的限制，擴展了 BLE 的應用範圍，特別適用於無線感測網路（Wireless Sensor Networks, WSN）和物聯網（IoT）環境。

Bluetooth Mesh 保有 BLE 的優點，並改善 BLE 傳輸範圍有限的問題，Bluetooth Mesh 的網路架構分為 Flooding 模式及 Routing 模式，Flooding Mesh 是大多數 BLE Mesh 協議採用的傳輸模式，利用廣播方式傳送訊息，無需建立節點間的連接，節點將訊息以廣播模式傳遞給通訊範圍內的所有節點，節點收到訊息後再次廣播，直到訊息傳遞至目標節點為止。

Routing Mesh 使用連接模式，節點需在訊息傳輸前建立節點間的連接，確保訊息傳遞有序且可靠。節點之間先建立連接，資料通過預設路徑逐步傳遞至目的節點，並根據需求動態調整路由，因為資料傳輸通過固定路徑，減少碰撞和干擾，讓傳輸更穩定。也因為訊息傳遞通過固定的路徑，降低重複廣播次數，節省了不必要的電能消耗。Routing Mesh 資料傳輸的方式確保資料完整傳遞，即使網路繁忙也能正常運作。相對的缺點也顯而易見，因為資料傳遞時，節點間需建立連接，設計上比較複雜而增加實現難度。且建立連接需耗費額外時間，當路徑上的節點完成所有連接後才會開始傳輸，所以網路啟動比較慢。也會因為接點間連接路徑較長，讓資料傳遞時間增加，最後產生較高的傳輸延遲。

2.3 分時多工與 BLE 中的 CI 與 CE 機制

分時多工（Time Division Multiple Access, TDMA）如字面上的意義，它一種將時間分割的通訊技術，讓多個裝置可以高效的共享同一傳輸介質進行傳輸。TDMA 將時間劃分為多個時槽（Time Slots），每個裝置只能在指定的時槽內進行傳輸，這確保了同一時刻，只有一個裝置進行傳輸，其他的裝置在同一個時間下（相同 Time slot），都處於等待的狀態，可以確保訊號穩定且不重疊，有效避免了訊號重疊和干擾而導致的數據丟失。

在 BLE 中，TDMA 的理念被具體實現在連線模式下的 CI 與 CE 設計中。BLE 通訊雙方在建立連線後，會協商出一個固定的 CI，作為接下來通訊週期的基本單位。每個 CI 之內，雙方僅在特定的 CE 中進行資料交換，其餘時間則處於睡眠或非活躍狀態。

這樣的設計與 TDMA 的概念密切相關：每個 BLE 連線設備實際上都在預定的時槽（即 CE 時間內）進行資料交換，彼此間在時間上分離，以避免干擾與資源衝突。尤其在多裝置共存環境中，例如一個主裝置（Central）與多個從裝置（Peripheral）連線時，主裝置會根據各連線的 CI 設計，輪流與不同的裝置進行資料交換，實現一種類似 TDMA 的排程與存取控制機制。

2.4 目的地傾向傳輸

在 FruityMesh 的傳輸機制中，訊息從一個節點發送至中繼節點時，中繼節點會廣播訊息給所有相鄰節點，相鄰的節點又會在廣播給其相鄰的節點，直到目標標節點，而廣播風暴會造成節點間多餘不必要的資料傳輸，而增加網路的負擔。為了解決廣播風暴問題，目的地傾向傳輸（Destination-Oriented Transmission, DOT）[2] 機制，通過引入方向性和目的傾向的傳輸策略，優化封包的傳輸過程，減少網路負載與封包重傳，提升傳輸效率與穩定性。

DOT 機制利用樹狀架構中的 Level 層級概念，為每個節點計算其所屬層級，使用 FIND_LEVEL 封包進行層級標記，傳輸方向封包只需傳遞至 Level 較低的節點，避免無意義的廣播。DOT 在中繼節點限制封包的傳輸方向，僅向 Level 較低的節點傳輸封包，這種方式有效減少了網路流量，降低碰撞率，並提升了傳輸可靠性。

2.5 目的地傾向切換傳輸

目的地傾向切換傳輸（Destination Oriented Switch Transmission, DOST）[2] 機制進一步解決角色身份重疊與傳輸重疊的問題，通過啟用與禁用連線的方式，降低網路負載並避免封包重傳，解決 BLE 網狀網路節點可能因同時扮演 Master 和 Slave 角色衝突而導致的封包傳輸遺失現象。

啟用與禁用功能，節點傳輸 INIT_STATE 封包，用於標記和控制各連線的啟用或禁用狀態，確保所有節點對於連線的處理具有一致性，節點在某一連線傳輸封包時，禁用其他連線以避免通訊重疊，每個 Connection Interval 結束後，切換連線的啟用與禁用狀態，雖然可能導致封包延遲一個 Connection Interval，但有效降低了網路負載，提升傳輸效率，如圖2.6所示。

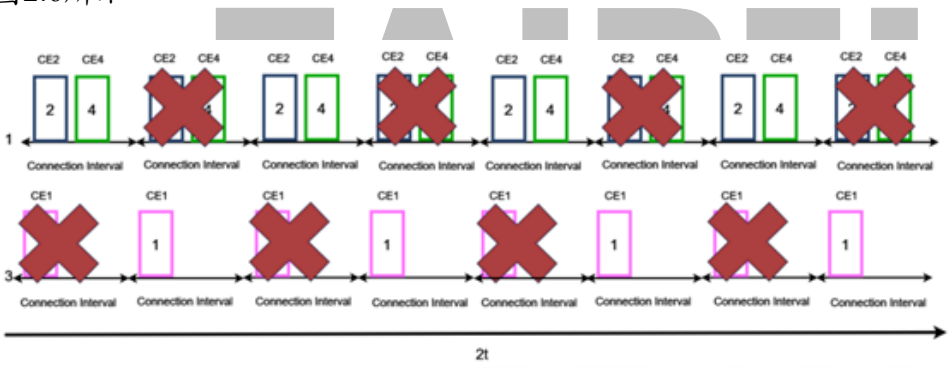

圖 2.6 傳輸機制的節點時序示例 [2]

第三章　BLE Mesh 拓樸建立機制設計

3.1　問題分析

3.1.1　BLE Mesh 拓樸建立問題

在原生 FruityMesh 架構中，節點的建立與連線並未針對資料匯集或匯出端進行特別設計，因此整體網路並無明確的 Root 節點。其封包傳輸方式主要採用廣播機制，即每當節點產生封包時，會向所有相鄰節點廣播傳送，期望藉由鄰近節點的重傳將封包推進至目的地。雖然此方法具備一定的自我修復能力，但在多節點、大範圍的 Mesh 網路中，極易導致封包重複傳送與碰撞，進而產生所謂的「廣播風暴」現象，嚴重影響整體網路效能與穩定性。

為了解決此問題，研究中引入 DOT 機制 [2]，將原先的無向廣播傳輸改為目的導向的封包路由方式，透過樹狀拓樸的建立使資料傳輸路徑更具方向性與效率性。然而，導入 DOT 排程機制的前提之一，是整個 BLE Mesh 網路需具備清楚的階層結構，而 Sink 節點（資料匯集端）必須作為整體樹狀拓樸的根節點。唯有如此，封包才能沿著既定的父子節點路徑，自節點有效匯流至 Sink 節點，達成 DOT 排程所需的單向、無冗餘傳輸目標。

然而，由於 FruityMesh 原生設計並非以 Sink 節點為中心的拓樸為出發點，現有拓樸建立流程尚無法保證 Sink 節點必然成為根節點。在未進行拓樸控制調整的情況下，可能出現 Sink 位置處於樹葉節點甚至中繼節點等非理想情境。因此，若要在 FruityMesh 架構中有效實作 DOT 排程機制，勢必需重新設計拓樸建立邏輯，強制指定使 Sink 節點始終位於拓樸樹的根部，即使在斷線並重新連線的情況下，亦能恢復其根節點角色，確保網路穩定性與傳輸效能。

3.1.2　BLE Mesh 封包傳輸問題

在基於 FruityMesh 的藍牙低功耗網狀網路傳輸機制設計中，[2] 研究已針對封包延遲與封包抵達率等品質指標進行優化，並取得初步成效。該研究藉由導入 DOT 與 DOST 排

程機制，成功改善了大部分的傳輸延遲與資料完整性。然而，即使在優化機制下，網路中仍不時出現封包遺失（掉封包）與重傳現象，顯示目前的傳輸機制尚存在進一步提升的空間。

造成封包掉落與重傳的關鍵因素之一，在於 BLE Mesh 網路的流量集中特性。由於整體網路皆以 Sink 節點作為封包的最終目的地，所有資料封包最終皆會匯集至該節點，導致越接近 Sink 的節點需承擔更多的中繼與轉發任務。這種負載不均的現象將使中樞節點在短時間內接收到大量傳輸要求，導致中樞節點的 Sent buffer 資源迅速耗盡。一旦緩衝區爆滿，不僅無法即時處理新進封包，還可能造成排程延遲、封包掉落，進而引發重傳機制啟動，進一步加重網路負載。

為有效緩解此類壅塞現象，需從連線參數層級進行調整，特別是針對 CI 與 CE 進行優化配置。透過適當調整 CI 間隔，可有效控制節點可傳輸封包的節奏與頻寬分配，進一步降低高負載節點的壓力，提升整體封包處理效率。結合連線參數調整機制，可以有效改善封包壅塞問題，以提升 BLE Mesh 網路在多節點、高流量情境下的服務品質。

3.2 BLE Mesh 拓樸建立機制設計

本論文提出改善 FruityMesh 的拓樸建立機制，確保 Sink 節點始終位於整體網路的根節點，並在斷線後能夠自動恢復其角色。此設計為後續的 DOT 排程機制提供穩定的基礎，如圖3.1。

在拓樸建立流程啟動時，每個節點在開機後會廣播包含自身節點資訊的 JOIN_ME 封包。節點收到其他節點所廣播的 JOIN_ME 封包後，會將該封包儲存於本地緩衝區中，並根據封包內容進行評分（scoring），以評估對方作為連線對象的適合程度。此評分機制可根據鄰近程度、RSSI 訊號強度、是否為 Sink 節點等參數進行加權計算，從而判斷潛在連線對象的優先順序。

若節點本身為預先指定的 Sink 節點，則會主動嘗試以 Master 的角色發起連線，以確保其成為拓樸的根節點。另一方面，對於非 Sink 節點，為了確保拓樸架構的根節點為 Sink 節點，節點在評估過程中會優先以 Slave 身分尋找合適的 Master 進行連線。只有當未能找到合適的 Master 節點（例如評分皆為 0 或未收到有效的 JOIN_ME 封包）時，該節點才會轉而作為 Master 嘗試連線至其他尚未建立連線的節點，進一步擴展網路的拓樸

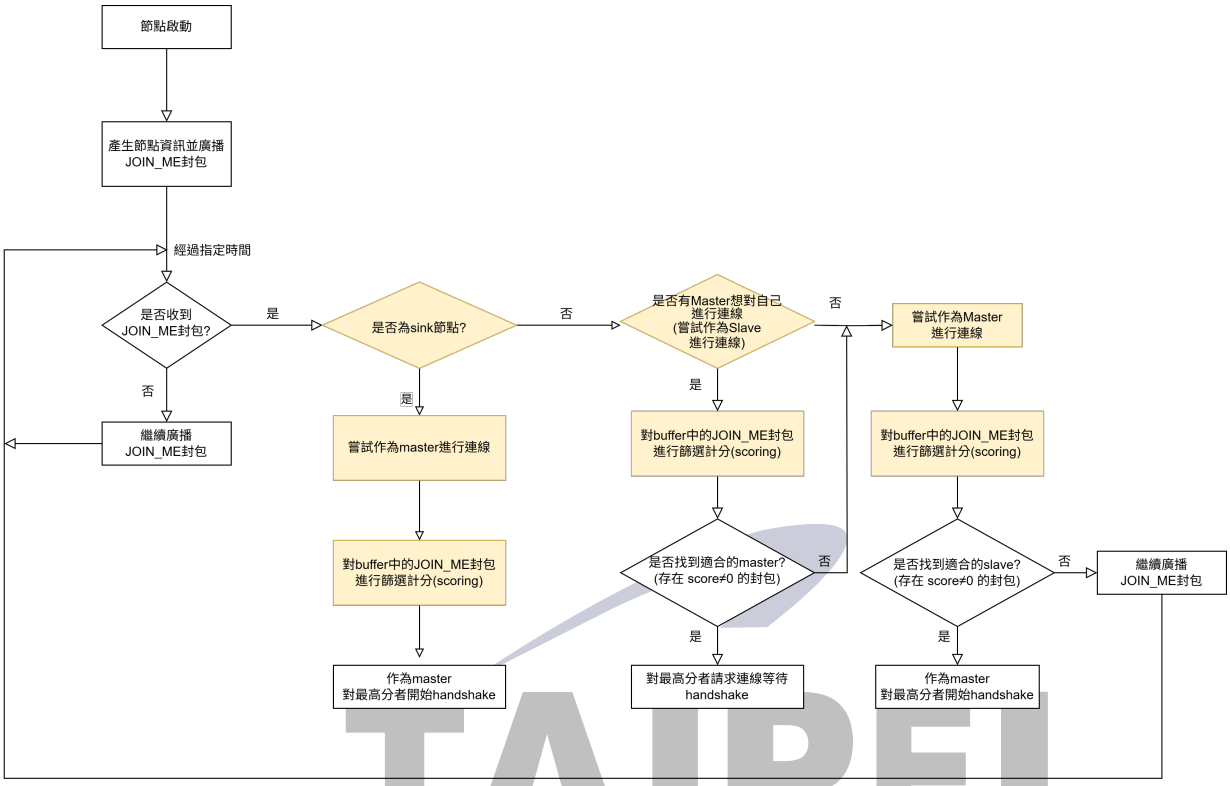

圖 3.1 提出的拓樸建立流程圖

結構。

若節點在轉為 Master 之後，仍未能找到合適的 Slave 節點（即緩衝區中所有 JOIN_ME 封包的評分皆為 0 或無法建立穩定連線）時，該節點將回到初始狀態，繼續定期廣播 JOIN_ME 封包，以等待後續更合適的連線機會出現。此重試機制能避免節點因短期內找不到連線對象而陷入孤立狀態，提升整體拓樸建立的成功率與網路的自組織能力。

3.2.1 決定最好的群組加入

Algorithm1描述了節點在藍牙低功耗網狀網路（BLE Mesh）中，決定其與其他節點建立連線方式的邏輯流程。其核心目標是依據當下網路狀態與節點角色，判斷該節點應該主動發起連線（作為 Master）或是被動接受連線（作為 Slave）。

流程一開始會檢查當前節點是否為 SINK 且具備可用的對外連線資源。若節點為 Sink 節點，則優先考慮作為 Master 嘗試連線，並透過 DetermineBestClusterAsMaster() 函式選出最合適的連線目標節點。若成功取得候選節點，系統會根據對方裝置的類型調整連線參數（如 CI），並嘗試建立連線。若連線成功，系統會更新連線時間與次數統計，

Algorithm 1 DetermineBestClusterAvailable Function

```
Initialize result as NO_NODES_FOUND
Get device type and assign to deviceType
if deviceType is SINK and outbound connections are available then
    bestClusterAsMaster ← DetermineBestClusterAsMaster()
    if bestClusterAsMaster ≠ null then
        Adjust connectionIv based on peer' s device type
        Attempt to connect as Master
        if connection succeeds then
            Update connection attempt time and count
        end if
        Set result.result ← CONNECT_AS_MASTER
        Set result.preferredPartner ← bestClusterAsMaster.sender
        return result
    end if
else
    Reset currentAckId to 0
    bestClusterAsSlave ← DetermineBestClusterAsSlave()
    if deviceType is SINK then
        bestClusterAsSlave ← null
    end if
    if bestClusterAsSlave ≠ null then
        currentAckId ← bestClusterAsSlave.clusterId
        if meshMaxInConnections == 1 then
            Check if any fresh connection exists (handshake not expired)
            if no fresh connection and freeMeshInConnections == 0 then
                if clusterSize ≠ bestClusterAsSlave.clusterSize OR random trigger passed then
                    Force disconnect other mesh connections
                    Reset cluster size to 1
                    Generate new clusterId
                end if
            end if
        end if
        Update JoinMe packet
        Set result.result ← CONNECT_AS_SLAVE
        Set result.preferredPartner ← bestClusterAsSlave.sender
        return result
    end if
    bestClusterAsMaster ← DetermineBestClusterAsMaster()
    if bestClusterAsMaster ≠ null and outbound connections are available then
        Adjust connectionIv and attempt to connect
        if connection succeeds then
            Update connection info
        end if
        Set result.result ← CONNECT_AS_MASTER
        Set result.preferredPartner ← bestClusterAsMaster.sender
        return result
    end if
    Log "no cluster found"
    Set result.result ← NO_NODES_FOUND
    return result
end if
```

並回傳連線成功的決策結果。

若節點角色並不是 Sink 節點，則會優先作為 Slave 接受他人連線。此時會重設回應欄位（currentAckId），並透過 DetermineBestClusterAsSlave() 選出合適的候選節點。並再次檢查當前節點如果為 SINK 節點，則不允許 Sink 節點作為 Slave 進行連線，。當節點作為 Slave 進行連線時，若符合條件的 Master 節點存在，系統則會建立連線。當節點作為 Slave 進行連線沒有找到適合的 Master 時，則該節點會再度嘗試以 Master 身分建立連線，重複執行候選節點評估與連線流程，若成功，則回傳連線結果。最後如果所有嘗試皆沒有辦法成功連線，系統會記錄「未找到叢集」的訊息，並回傳 NO_NODES_FOUND 的決策結果。

3.2.2 以 Mater 身分選擇最佳的 Slave

Algorithm 2 CalculateClusterScoreAsMaster

```
Retrieve device type from device configuration
if packet is too old then return 0
end if
if packet.clusterId == this.clusterId then return 0
end if
if packet has no free inbound connections then return 0
end if
if packet.ackField ≠ this.clusterId and ≠ 0 then return 0
end if
if packet.clusterSize > current cluster size and current device is not SINK then return 0
end if
if packet is temporarily blacklisted then return 0
end if
if already connected to packet.sender then return 0
end if
if packet.rssi < threshold then return 0
end if
if current device type == LEAF then return 0
end if
rssiScore ← 100 + packet.rssi
score ← packet.freeMeshOutConnections + rssiScore − (packet.hopsToSink × 1000)
Modify score using preferred partner policy
return score
```

Algorithm2說明了節點在嘗試成為 Master 時，如何針對鄰近的叢集候選節點進行評分。該評分機制用來決定是否值得主動與某個節點建立連線。

一開始，節點會先取得自身的裝置類型，若為 LEAF（葉節點）則無法發起連線，直接回傳 0 分。接著系統會篩選掉以下不合適的候選節點：如封包太舊、叢集已經相同、

對方無 inbound slot、ack 欄位不符、對方叢集較大（非 SINK 狀態下）、對象短時間內已多次連線失敗（暫時黑名單），或已與該節點連線中等情形。此外，如果 RSSI 過低，也會被排除。

通過初步篩選後，系統會以 RSSI 為基礎計算連線品質分數 rssiScore，再根據對方節點的連線資源與至 Sink 節點的跳數（hopsToSink）進行加權計算整體分數。該分數越高，代表對方節點越適合作為連線對象。

最後，系統還會根據偏好節點（preferred partner）進行微調後，回傳最終評分結果。

3.2.3 以 Slave 身分選擇最佳的 Master

Algorithm 3 CalculateClusterScoreAsSlave

Retrieve device type from configuration
if deviceType == SINK **then return** 0
end if
if packet.hopsToSink < 0 **then return** 0
end if
if packet.freeMeshOutConnections == 0 **then return** 0
end if
if packet is too old **then return** 0
end if
if packet.clusterId == current.clusterId **then return** 0
end if
if packet.clusterSize < current cluster size **and** packet.deviceType $\neq$ SINK **then return** 0
end if
if packet.rssi < threshold **then return** 0
end if
$rssiScore \leftarrow 100 + packet.rssi$
$score \leftarrow 0$
if packet.deviceType == SINK **then**
    $score \leftarrow score + 10000$
end if
$score \leftarrow score + (packet.hopsToSink \times 1000) + (packet.clusterSize \times 100) + (packet.freeMeshOutConnections \times 100) + rssiScore$
Modify score based on preferred partner logic
return score

Algorithm3描述了節點在考慮作為 Slave 加入其他叢集時，如何針對潛在的 Master 節點進行評分。

首先系統會排除不適合的情況：若目前節點為 SINK、候選節點無 outbound slot、封包太舊、對方為同一叢集、或是對方叢集規模小於本地叢集且並非 SINK，皆會被排除，且 RSSI 資訊不穩定，也不納入考量。

若符合條件，則會計算 RSSI 分數，並依據以下項目計算總分：

- 候選節點是否為 SINK（可給予額外權重）
- 對方距離 Sink 的跳數（hopsToSink）
- 對方叢集大小（clusterSize）
- 可用的 outbound slot 數量（freeMeshOutConnections）
- 實際 RSSI 品質分數（rssiScore）

最終分數會再次根據偏好節點機制進行修正，作為節點選擇最佳 Master 的依據。

3.2.4 Self-Healing 機制的拓樸修復改進

原生 FruityMesh 提供了 Self-Healing 機制，使節點在斷線後能回到 HIGH_DISCOVERY 狀態並尋找可連線的鄰居節點，從而重新建立連線並合併至單一 Cluster。然而，此機制下若斷線的是 Root（Sink）節點，重新連回後將失去 Root 身份，可能導致整個網路拓樸失效或性能下降。

為解決此問題，我們修改了 CLUSTER_WELCOME 訊息的握手流程。在握手判斷邏輯中，若發現對方為 SINK 且其 Cluster 比自己小（或等於），則避免讓原本為 Root 的節點退位為普通節點，並強制原 Cluster 切斷其他連線，重新啟動拓樸建立，使原本的 SINK 節點能保有 Root 身份，維持拓樸穩定性與網路一致性，如 Algorithm4。

Algorithm4為了強化 FruityMesh 在節點斷線重連後的拓樸穩定性，本研究在原有 Self-Healing 機制之上，進一步修改其握手邏輯。當節點收到 CLUSTER_WELCOME 封包後，將進入拓樸重組的判斷流程。具體邏輯如下：

首先，節點會確認封包的大小是否正確，若不正確則忽略該封包。接著，進入 Handshaking 狀態並備份目前的 Cluster ID 與 Cluster 大小，用以後續比對使用。

若對方節點的 Cluster ID 與自己相同，代表已屬同一個 Cluster，這在初始階段是不應該出現的情況，因此直接斷線處理。若對方的 Cluster 大小比本地端小，且其並非 SINK 節點，則本節點認定自己應作為主導方，若該連線是 inbound，則代表方向錯誤，同樣會中止連線。

此外，若兩節點的 network ID 不一致，或連線對象並非偏好的節點（根據配置設定），也將被強制中止以避免不必要的 Mesh 錯誤連線。

在以上檢查皆通過的情況下，代表本節點應接受對方節點加入自己的 Cluster。若本節點為 SINK 節點，則會在 CLUSTER_ACK_1 回覆中指定 hopsToSink = 0，表明自己為 Mesh 根節點，並強制中斷其他 Mesh 連線，重新以自己為中心建立拓樸結構，確保在重新連線的情況下仍能維持 SINK 節點的 Root 地位。

這套改進邏輯可大幅提升 BLE Mesh 在 SINK 斷線後的自我修復能力，並維持以 SINK 為 Root 的拓樸一致性。

Algorithm 4 Topology Self-Healing Handshake Logic

Require: CLUSTER_WELCOME packet received

```
if packet size invalid then
    Log and ignore packet
else
    Save partner handle and enter HANDSHAKING state
    Backup local cluster ID and size
    if remote cluster ID == local cluster ID then
        Disconnect: SAME_CLUSTERID
    else if remote cluster size < local size and remote is not SINK then
        if connection is inbound then
            Disconnect: WRONG_DIRECTION
        end if
    else if network ID mismatch then
        Disconnect: NETWORK_ID_MISMATCH
    else if connection is not preferred and preferences ignored then
        Disconnect: UNPREFERRED_CONNECTION
    else
        Accept connection as Root
        Send CLUSTER_ACK_1 with hopsToSink = 0 if self is SINK
        Disconnect all other mesh connections
        Reinitialize local cluster with size = 1 and new ID
    end if
end if
```

參考文獻

[1] Mordor Intelligence. *Internet of Things (IoT) - Market Share Analysis, Industry Trends & Statistics, Growth Forecasts 2024–2029*. Accessed: 2025-06-04. Jan. 2024. URL: https://www.gii.tw/report/moi1403099-internet-things-iot-market-share-analysis-industry.html.

[2] 温淇淼. "基於 FruityMesh 之藍牙低功耗網狀網路傳輸機制的設計以改善 QoS". PhD thesis. 台北市, 2024, p. 41. URL: https://hdl.handle.net/11296/7u74xv.

[3] 吳俊諺. "基於 FruityMesh 之 BLE TDMA 傳輸機制的設計與實現". PhD thesis. 台北市, 2022, p. 48. URL: https://hdl.handle.net/11296/a3g254.

[4] 郭家瑋. "基於 Nordic SoftDevice 之 BLE TDMA 傳輸機制的設計與實現". PhD thesis. 台北市, 2021, p. 55. URL: https://hdl.handle.net/11296/5c352v.

[5] MICROCHIP Developer Help. *Bluetooth Link Layer*. https://microchipdeveloper.com/xwiki/bin/view/applications/ble/introduction/bluetootharchitecture/bluetooth-controller-layer/bluetooth-link-layer. Accessed: Nov. 2023. Nov. 2023.

[6] ángela Hernández-Solana et al. "Bluetooth Mesh Analysis, Issues, and Challenges". In: *IEEE Access* 8 (2020), pp. 53784–53800. DOI: 10.1109/ACCESS.2020.2980795.

[7] Bluetooth SIG. *Bluetooth Core Specification Version 5.0*. https://www.bluetooth.com/specifications/specs/core-specification-5-0/. Accessed: 2025-06-04. 2016.

[8] Eunjeong Park, Hyung-Sin Kim, and Saewoong Bahk. "BLEX: Flexible Multi-Connection Scheduling for Bluetooth Low Energy". In: *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (Co-Located with CPS-IoT Week 2021)*. IPSN '21. Nashville, TN, USA: Association for Computing Machinery, 2021, pp. 268–282. ISBN: 9781450380980. DOI: 10.1145/3412382.3458271. URL: https://doi.org/10.1145/3412382.3458271.