Multi-Connection Scheduling for Resource Fairness in Bluetooth Low Energy Networks

Moonbeom Kim and Jeongyeup Paek
Department of Computer Science & Engineering, Chung-Ang University, Seoul, Republic of Korea
{mbkim, jpaek}@cau.ac.kr

***Abstract*—Bluetooth Low Energy (BLE) is positioned as a representative wireless technology in Internet of Things (IoT) applications and systems. It enables communication with multiple device at low-power and includes diverse features for providing various services with high-quality. However, the Bluetooth specification does not specify the resource management and scheduling mechanisms for multiple connections. Since each connection is independently handled, achieving optimized schedule is an important research topic in Bluetooth networks. To address these challenges, we propose a *"Fair Multi-Connection Scheduling" (FM-Schedule)*, which schedules and allocates resources for new connections taking into account the requirements of previously connected peripherals. We implements *FM-Schedule* on real embedded devices, and evaluate its performance compared to popular BLE stacks (i.g. NimBLE, Zephyr OS, and Nordic) in real environment.**

***Index Terms*—IEEE 802.15.1, Bluetooth Low Energy (BLE), Multi-Connection Scheduling, Resource Fairness.**

I. Introduction

Bluetooth Low Energy (BLE), a representative wireless technology for low-power and multi-connection, is pervasive in our daily lives. Bluetooth is commonly embedded in smartphones, desktops, tablets as well as various everyday devices for wireless audio/video, vehicular, environment sensing or control, etc. Furthermore, to provide low-power network services, it is widely adopted in many Internet of Things (IoT) systems including smart home, smart factory, and smart market.

The Bluetooth SIG has been steadily improving Bluetooth technology for several years and is constantly working to advance it [1]. Although the latest Bluetooth specification defines diverse features[1] for improving performance and providing high-quality services, it does not specify how the central (a.k.a. master in BLE) schedules and allocates resources of each peripheral (a.k.a. slave in BLE), which is independently handled [3]. Resource scheduling is a very important and essential mechanism for suitable and fair resource allocation

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1A4A5034130 & No. 2021R1A2C1008840), and also by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2023-RS-2022-00156353) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

[1]Such as multi-connection, mesh network, LE audio [2], channel classification, advertising enhancement, etc.

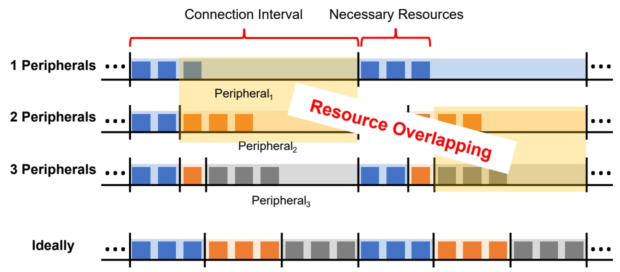


Fig. 1: Resource Overlapping Problem.

to peripherals depending on their required Quality of Service (QoS). Especially, central BLE, which establishes and manages multiple connections, has to schedule the connection of each peripheral, allocating the appropriate resources. Nevertheless, due to the lack of specifications as mentioned above, integrating and timely scheduling each connection event becomes a very complex challenge.

The popular Bluetooth stacks such as NimBLE [4], Zephyr OS [5], and Nordic [6] apply their own mechanisms to schedule the multiple connections. However, these methods are very simple and straightforward, and each connection is still scheduled and managed independently. None of them have considered the requirements and status of previously connected peripherals when establishing the new connection. It causes the *resource overlapping problem*, where some other connection invades the resources allocated for existing connection. This problem results in not only performance degradation, but also disconnection between central and peripheral. For example, suppose that $peripheral_1$ is already connected to $central$, and that $peripheral_2$ attempts to connect with the same terms as $peripheral_1$ (connection interval and necessary resources), as shown in Fig. 1. Then, the $central$ decides the *anchor point* to start communication for $peripheral_2$ without consideration anythings for $peripheral_1$, and this ultimately leads to the *resource overlapping problem.*

To address this problem, we propose *"Fair Multi-Connection Scheduling" (FM-Schedule)*, which determines the anchor point at the right point on the timeline to schedule and allocate appropriate resources. Furthermore, it consolidates each connection that is handled independently, managing them into a single timeline. In this work, we implement *FM-Schedule* on an nRF52840 DK [7] and evaluate performance

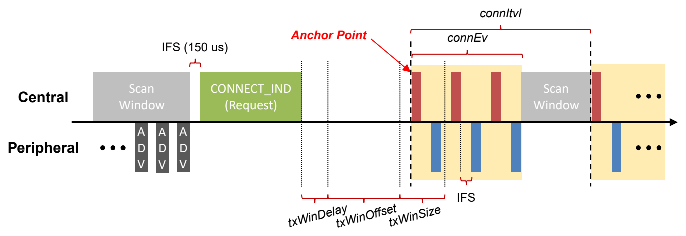


Fig. 2: Connection Establishment Procedure

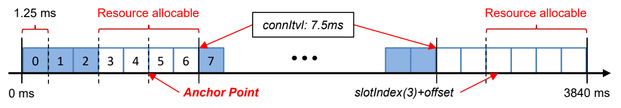


Fig. 3: FM-Scheduleoverview

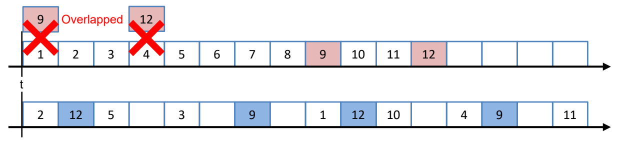


Fig. 4: Overlapping between peripherals with different connItvl

by comparing it to the popular BLE stacks (i.e. NimBLE, Zephyr OS, and Nordic) in real environments.

II. Background

A. Connection Establishment

Fig. 2 illustrates how connection is established. BLE devices have two roles the central and peripheral. To establish a connection, the peripheral broadcasts advertising (*ADV*) packets using three advertising channels (ch. 37, 38 and 39), and the central scans these channels sequentially. When the central receives the ADV packet, it sends the CONNECT_IND (i.e. connection request) to the peripheral, after inter frame space (IFS). The CONNECT_IND includes several connection parameters such as transmit window delay (*txWinDelay*), offset (*txWinOffset*), size (*txWinSize*), connection interval (*connItvl*), and so on. Transmit window parameters such as *txWinDelay*, *txWinOffset* and *txWinSize* affect the decision of *anchor point* that is the time of connection event (*connEv*) start for data exchanging. The *txWinDelay* is fixed at 1.25 ms when PHY is 1 Mbps, and the *txWinOffset* defines the beginning of the transmit window for transmitting the first data packet from the central.

If the central sent the CONNECT_IND and the peripheral accepted it, after a specific time (*txWinDelay* + *txWinOffset*), the central transmits the first packet within the transmit window (i.e. *txWinSize*), while the peripheral stays in the first data channel to listen for it. When the connection is established, the central and peripheral wake up and exchange packets periodically per *connItvl* on the 37 data channels. The length of *connEV* actually represents the amount of resources available to transmit packets between the central and peripheral pair. Every *connEv* is started with the transmission of a data or empty packet by the central. If central and peripheral have no more data to transmit, the *connEv* is finished.

B. Resource Scheduling of Popular BLE

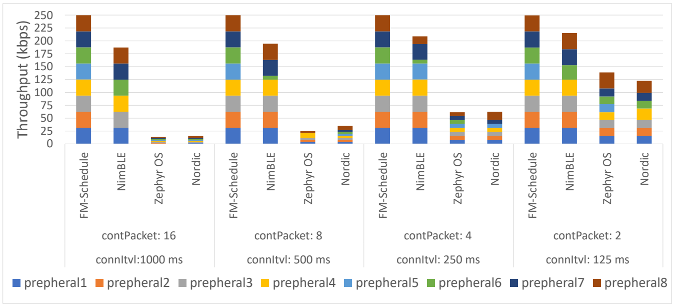
As shown in Fig. 2, the central usually resumes the scanning process when no packets are being exchanged. If the central scans the ADV packet of another BLE device during this time, it starts the procedure to connection. However, without any mechanism for anchor point decision or scheduling, this can cause the *resource overlapping problem* as shown in Fig. 1. The central finishes the current *connEv* (for $peripheral_1$) to start the next one (for $peripheral_2$) because there has to be only one active *connEv* at a time. For this reason, the current *connEv* is blocked, and its length is shorter than the time needed to exchange the packets.

Popular BLE stacks have their own algorithm to make anchor point decision and a priority policy for scheduling. When the centrals in the most popular BLE stacks receive an ADV packet from a peripheral, they want to establish the new connection as soon as possible. Because of this, they all set the default value of *txWinOffset* to zero and attempt to schedule a new connection; *txWinSize* is set to the smallest possible value that does not affect synchronization and allows for rapid transmission of the first packet (NimBLE: 2 [4], Zephyr OS and Nordic: 1 [5], [6]). Then, the centrals check if the new connection overlaps with the current one. In case of overlap, they check whether it can preempt an existing connection, using their own priority policy. If it cannot preempt, the centrals move the anchor point of new connection past the current connection by increasing the *txWinOffset* and *txWinSize*. The centrals repeat this process and verify whether the new connection is still within the allowed range.

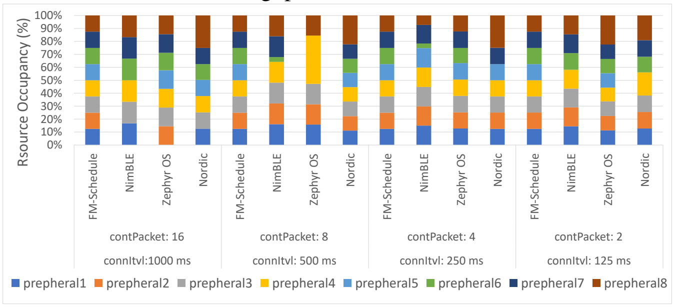
III. Design

We suppose that the central knows how much data it will transmit from each peripheral, and that each packet arrives within a three transmission opportunities, including retransmissions. The *FM-Schedule* allocates the resources for peripherals on a *resource list* that consists of 3072 small slots, enabling the integrated management of each connection. The reference time in the *resource list* is clock of central and has a period of 3.84 seconds, as shown in Fig. 3. Each slot represents a 1.25 ms time resource.

The size of the list and scheduling complexity are increased exponentially, if the each *connItvl* is mutually prime. To solve this, we leverage the approach in BLEX [8]. The *FM-Schedule* recalculates the *connItvl* to $7.5 * 2^n$ ms $(0 \leq n \leq 9)$, close to the original *connItvl*, before it transmits the CONNECT_IND. Then, to find the maximum empty slots, it searches sequentially the *resource list* from the *firstSlotIndex(0)+offset* to the *connItvl/1.25 ms+offset*; the *offset* equals to *connItvl*k* $(0 \leq k \leq$ *listLenght/connItvl*). If the empty slots is more than twice as large as time to successfully transmit all packets, *FM-Schedule* determine the anchor point as index I ($I = \lfloor \frac{(lastEmptyIndex - firstEmptyIndex)}{necessaryResources * 2} \rfloor * necessaryResources$). Otherwise, the anchor point is set to the *firstEmptyIndex*. The reason



(a) Total throughput measured at the central.



(b) Resource occupancy for each peripheral in total throughput.

Fig. 5: Performance results for each scheduling mechanism in the multiple connections scenarios under different conditions

is to avoid overlap between peripherals with different *connItvl*, as shown in Fig. 4. For example, as shown in Fig. 3, if there are 4 slots available for resource allocation and 2 slots are required for data transmission, the anchor point of the new connection is set to the start time of index 5 slot.

IV. Evaluation

We evaluate the performances for throughput and fairness of the *FM-Schedule* through comparative experiments with NimBLE, Zephyr OS and Nordic. We implement the *FM-Schedule* and the other stacks on a real embedded device (nRF52840 DK by Nordic Semiconductor [7]). Our BLE network consists of one central and seven peripheral, and each peripheral is sequentially connected to the central. We conduct the experiments in uplink scenarios with different connection interval (*connItvl*) and the number of packets transmitted continuously (*contPacket*). *connItvl* ranges from 1000 ms to 125 ms, and *contPacket* ranges from 16 to 2. In each scenario, *connItvl* and *contPacket* are halved, and all peripherals have the same conditions for both. The data packet size transmitted by peripherals is set to the maximum (244 bytes).

Fig. 5 plots the throughput and fairness performances for *FM-Schedule* and each scheduling mechanism. The maximum reachable throughput of each peripheral is 31.232 kbps, and its total for eight peripherals is 249.856 kbps, in our all scenarios. The throughput results in Fig. 5a show that the throughput of *FM-Schedule* achieves the expected total throughput in all cases (avg. $\approx$ 249.797 kbps). The average throughput for peripherals is measured as $\approx$ 31.225 kbps. On the other hand, NimBLE has the maximum throughput of $\approx$ 215.11 kbps, which is $\approx$ 84 % of the expected result. For Zephyr OS and Nordic, the maximum throughputs are measured as $\approx$ 138.833 and $\approx$ 122.455 kbps, respectively. The minimum throughputs for three BLE stacks are $\approx$ 187.242, $\approx$ 13.508, and $\approx$ 15.502 kbps, respectively. These values are $\approx$ 25 %, $\approx$ 95 %, and $\approx$ 96 % lower than expected total throughput.

Furthermore, the *FM-Schedule* fairly allocates resources to peripherals in proportion to their total throughput measured, as shown in Fig. 5b. For the other scheduling mechanisms, a resource overlapping problem occurs; for instance, peripheral4 (NimBLE), 6 (Zephyr OS), and 8 (Nordic) in *connItvl* 500 ms and *contPacket* 8 scenario. Moreover, in the worst case, some peripheral is disconnected due to transmit queue overflow in all scenarios exclude *FM-Schedule*, as shown in Fig. 5b. This is because the central attempts to establish a new connection as soon as possible by preempting pre-scheduled connections based on a priority policy, without considering the current status of the peripherals. This can occur in case that there is short connection event, such as rare occurrences where all transmissions succeed without retransmissions.

V. Conclusion

In this paper, we have demonstrated that resource overlap occurs when using popular BLE stacks, and proposed a *Fair Multi-Connection Scheduling(FM-Schedule)* which is an effective resource scheduling scheme in multi-connection BLE networks. By managing the synchronized schedule with consideration of peripherals on a single timeline, *FM-Schedule* allocates fair resources to all peripherals. It achieves throughput up to $\approx$ 14 %, $\approx$ 44 %, and $\approx$ 51 % higher compared to other scheduling mechanisms. As our future work, we plan to improve the *FM-Schedule* for dynamic resource scheduling in various stress scenarios and conduct comparative experiments with prior studies [8], [9].

References

[1] "Bluetooth SIG (Special Interest Group)," [last accessed on July 2023]. [Online]. Available: https://www.bluetooth.com/

[2] D. Ryoo, Y. Yoo, J. Paek, and S. Bahk, "SPADE: Secure Periodic Advertising using Coded Time-Channel Rendezvous for BLE Audio," in *The 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, June 2023.

[3] "Bluetooth Core Specification Version 5.4," 2023, [last accessed on July 2023]. [Online]. Available: https://www.bluetooth.com/specifications/specs/core-specification-5-4/

[4] "Apache Mynewt-NimBLE," [last accessed on July 2023]. [Online]. Available: https://mynewt.apache.org/latest/network/index.html

[5] "Zephyr Project," [last accessed on July 2023]. [Online]. Available: https://www.zephyrproject.org/

[6] "Nordic Semiconductor," [last accessed on July 2023]. [Online]. Available: https://www.nordicsemi.com/

[7] "nRF52840 Development Kit," [last accessed on July 2023]. [Online]. Available: https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dk

[8] E. Park, H.-S. Kim, and S. 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, pp. 268–282.

[9] F. J. Dian, A. Yousefi, and S. Lim, "Time scheduling of central BLE for connection events," in *IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2018, pp. 763–767.