

A Runtime-Reconfigurable Wireless Body Area Network Using Textile-Based IoT Devices

Melike Beşparmak¹, Omur Fatmanur Erzurumluoglu¹, Kadir Ozlem¹, Asli Tuncay Atalay²,
Ozgur Atalay², Gökhan Ince¹

¹Faculty of Computer and Informatics Engineering, Istanbul Technical University, Istanbul, Türkiye

²Faculty of Textile Technologies and Design, Istanbul Technical University, Istanbul, Türkiye
{besparmak22, erzurumluoglu18, kadir.ozlem, aatalay, atalayoz, gokhan.ince}@itu.edu.tr

Abstract—Wireless Body Area Networks (WBANs) are an active research area with applications in health, sports, and entertainment domains. Yet, the literature offers few concrete implementations, forcing each new study to build a firmware stack from scratch. In this study, a runtime-reconfigurable WBAN framework over Bluetooth Low Energy is presented. Server devices stream sensor data through a dual-role hub that mirrors them onto a single client connection, so any application running on the client device is included in the network through that one connection. The framework is built on Zephyr real-time operating system and it runs across STM32WB, ESP32 variants, and nRF52840. Textile based IoT devices are integrated to demonstrate a real-world use-case. Connection interval, notification rate, and server count are reconfigurable at runtime. Baseline link performance, which is measured as packet loss and Round-Trip Time (RTT), is characterized across these three parameters. It is found out that the loss is caused by the hub-to-client link, and it accumulates as the offered load increases. RTT degrades together with loss, ranging from tens of milliseconds to several hundred under congestion. Additionally, contention becomes unfair as load rises, with some servers ceasing to deliver entirely. With real textile sensors, whose 50 Hz sampling rate bounds the packet rate, the network operates without loss. Thus, the congestion regime sheds light to the limitations of applications requiring higher-rate data processing.

Index Terms—Wireless body area network, wearable sensors, bluetooth low energy, textile based IoT devices, real-time operating system.

I. INTRODUCTION

Wireless Body Area Networks (WBANs) have long been an active research area, with existing work concentrated on surveys, protocols, and simulation studies [1], [2]. Despite their broad application space spanning health, sports, and entertainment, real-world deployments remain limited. Thus, any experimental research on WBANs requires a new implementation, costing time and effort. In this work, a framework adequate to be used with Textile-based sensors and actuators in IoT (T-IoT) devices is presented.

The framework is built on Zephyr Real-Time Operating System (RTOS) ¹ to enable interoperability across IoT and embedded technology product and service providers. The chosen wireless technology is Bluetooth Low Energy (BLE) ² since it

is low-power, widely available, and natively supported by computers and mobile phones. Three vendors (STM32WB, ESP32 variants, and nRF52840) were used during the experiments, demonstrating that the framework is portable when a new device is introduced. Two custom sensors have been integrated: a textile-based sensing glove and knee brace. Textile-based sensors are well suited for wearable technologies [3], [4], as they feel natural to wear. However, e-textile sensing work is often demonstrated as standalone sensing elements rather than being integrated into a complete, networked system operating in a real deployment [5]. The proposed framework integrates these sensors into a unified WBAN system. Since any custom sensor requires a different data format, the packet structure is designed to be flexible and self-describing. To accommodate the needs of different use cases having different demands, a runtime-reconfigurable framework is necessary. Therefore, the Connection Interval (CI), notification rate, and server count can be adjusted via the client device.

This paper makes the following contributions: (i) a BLE framework that connects multiple server devices and is reconfigurable during runtime, (ii) integration of textile-based IoT devices into the framework, (iii) a reproducible characterization of packet loss and Round-Trip Time (RTT) across connection interval, notification rate, and server count using wearable textile-based sensors.

II. RELATED WORK

Most of the WBAN literature consists of surveys and protocol designs rather than real implementations and performance evaluation. Existing surveys cover architectures, applications, and open issues, and consistently note that proposed solutions rarely meet real-world deployment requirements [1], [6]–[9]. Much of the remaining work relies on simulation rather than hardware, from analytical models of BLE throughput and latency versus connection interval [10] to simulation-validated clustering optimizations for WBAN-IoT [11]. The IEEE 802.15.6 standard [12] likewise defines a dedicated WBAN PHY/MAC, but most wearables use BLE in practice.

Real implementations remain comparatively rare, smart-healthcare prototypes mention BLE but report minimal link-level results [14]. Closer to our own setting, one study mitigates packet loss below 1% in a wearable ecosystem

This study was funded by the Scientific Research Project Unit of Istanbul Technical University (Grant No: MGA-2026-47385).

¹<https://www.zephyrproject.org>

²<https://www.bluetooth.com/specifications/specs/core-specification-5-4/>

by bundling data across a single peripheral pair [15], and another characterizes a BLE star network for prosthesis control across connection interval and server count, reporting that increasing connection interval from 10 ms to 100 ms grows maximum sustained connections from 4 to 20 [16]. SharkTooth provides firmware-free, application-layer time synchronization for multi-node BLE body sensor networks [17], and a related study identifies BLE's connection-interval floor as a constraint on multi-node body-worn systems relative to alternative low-power radios [18]. A cross-protocol benchmarking study on unified hardware similarly finds that connection interval governs BLE's stability/range trade-off, converging with our own finding that connection interval is the dominant factor in multi-node stability [19].

No prior work characterizes packet loss and latency across a systematic parameter sweep in a multi-node, multi-vendor, hub-aggregated WBAN. This work addresses that gap by providing detailed experimental results.

III. SYSTEM DESIGN AND IMPLEMENTATION

A. Topology and Server Representation

The proposed framework has three roles: server, hub, and client.³ Server devices carry the sensors, sampling and sending their data to the hub. The hub collects data from all servers and exposes it to the client, which hosts the application layer (a computer or mobile phone) and processes the incoming data. Fig. 1 shows how these roles map onto the network architecture.

Each server is a BLE peripheral that samples its sensors and exposes them over the Generic ATtribute Profile (GATT). The hub is a BLE central toward the servers and a BLE peripheral toward the client, making it the only path between them. The client (such as Python program or a Flutter application) connects to the hub by name.

Before the servers are accepted in the network, they advertise a network group ID. The hub scans, connects to matching advertisers, and runs a per-connection nonce challenge-response against a shared secret. It then records the verified peer in a flash-backed whitelist for directed reconnection. This avoids a hard-coded address list, where the hub firmware would need a rebuild for every new device.

The hub only mirrors the server devices. It forwards what it gets without processing the data. At boot, the hub registers one mirrored GATT service per server slot and supports up to ten slots. This limit is firmware dependent and can be updated, though the hardware of the hub device need to support that many simultaneous connections. A connecting server is assigned to the first free slot, and its service, together with its self-describing schema, is exposed through that slot's characteristics. As a result, the client distinguishes between servers and parses each one's data over the hub's single connection. This approach allows the client to be included in the network through a single connection. Thus, custom applications can easily be built on top of the hub.

³Code available at <https://github.com/metahead00/WBAN-as-a-Service>.

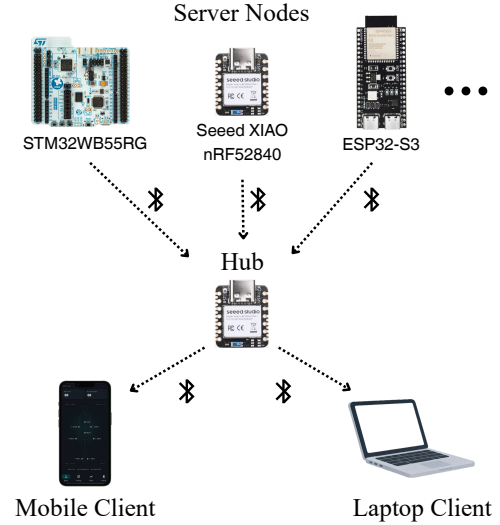


Fig. 1. The proposed architecture in the form of star topology, where all data flows through the hub.

TABLE I
HANDSHAKE PACKET FORMAT.

Field	Size (B)	Notes
ver	1	protocol version
cap_mask	2	advertised sensor capability mask
count	1	number of sensor entries
entries	variable	count $\times$ 3B entries: [type, dtype, n]

B. Self-Describing Sensor Packet

An IoT device may be equipped with more than one sensor, and different devices carry different sensors as seen in Fig. 2. As an example, our custom textile-based sensing glove [3] reports a fused orientation quaternion from an on-board absolute-orientation Inertial Measurement Unit (IMU) and five capacitive flex values, one per finger, whereas the textile-based knee brace [4] reports a single capacitive channel. Because any server device added in the future will expose a different mix of sensor types, value types, and value counts, a fixed packet layout would need editing for each new one. The packet format is therefore designed to be self-describing. Each server device declares its own layout once during handshake according to Table I. Then, every data streaming packet follows the schema in Table II and gets parsed according to handshake declaration.

1) *Handshake Packet Format*: On connecting, a server device sends a handshake record as a GATT indication, following the format in Table I: a protocol version byte, a 16-bit capability mask, an entry count, and three-byte entries per entry count, one per sensor the server device has, in ascending bit order. The capability mask reserves one bit per sensor type (e.g., accelerometer, gyroscope, temperature, ECG, capacitive, heart rate, resistive, quaternion), with nine types currently defined and seven bits reserved for future sensor types. Each

TABLE II
DATA STREAMING PACKET FORMAT.

Field	Size (B)	Notes
seq	2	
present_mask	2	
sensor_blocks	variable	decoded per handshake record
hub_rx_ts	4	appended by hub
hub_seq	2	appended by hub

entry then gives that sensor's data type and value count, so a sensor's block size is $width(dtype) \times n$. The BLE stack's indication confirm is itself the acknowledgment. The server device does not start streaming data until the hub confirms the handshake record, and it re-sends the record on every reconnect. After the handshake, both hub and client know exactly how to parse every packet from that server device. Table III(a) gives a worked example for our textile glove, CAP and QUAT entries stand for capacitive and quaternion sensor types.

Each entry in the handshake record additionally carries a one-byte data type field (INT16, INT32, or FLOAT32) giving that sensor value's width, and all fields are little-endian. One reserved bit in the `cap_mask` declares a synthetic BASELINE entry: when set, the server streams a fixed 50-byte payload instead of its physical sensors' data, so every server produces identical, comparable traffic regardless of its actual sensors.

2) *Data Streaming Packet Format*: Every notification then follows the format in Table II: a sequence number, a present-sensor mask, the sensor data blocks, and two suffix fields the hub appends before forwarding. `seq` counts sent packets only, so a gap is true loss. `present_mask` marks which sensors are fresh in this packet and is always a subset of `cap_mask`. A sensor sampled slower than the notification rate omits its block during that interval, so the record is variable length and is decoded by walking the handshake record in bit order, consuming $width \times n$ bytes per set bit. Before forwarding, the hub appends a 4-byte receive timestamp and a 2-byte monotonic forwarding sequence (`hub_seq`). Unlike the per-server `seq`, `hub_seq` is a single counter incremented over every record the hub relays across all servers, letting the client measure hub-to-client link loss directly. Table III(b) shows the corresponding data record for the glove example.

C. Runtime Reconfiguration over BLE

BLE connection interval, per-server notification rate, and the number of server devices are reconfigurable at runtime, with no reflashing and no interruption of the client's single connection to the hub. This allows the same firmware to be retuned for a different power/latency trade-off.

Configuration initially flows from the client to the hub. The hub then relays the new configuration to the server devices. The client writes a 7-byte runtime configuration record, which includes connection interval minimum and maximum, notification rate, and server count. The hub then updates

TABLE III
SELF-DESCRIBING PACKET, GLOVE EXAMPLE (LITTLE-ENDIAN).
OFFSETS/WIDTHS ARE RECOVERED FROM THE HANDSHAKE RECORD.

(a) Handshake packet format, (10 B).

Field	Offset (B)	Size (B)
ver	0	1
cap_mask	1	2
count	3	1
entry: CAP - INT16×5	4	3
entry: QUAT - FLOAT32×4	7	3

(b) Data streaming packet format, (36 B).

Field	Offset (B)	Size (B)
seq	0	2
present_mask	2	2
CAP - INT16×5	4	10
QUAT - FLOAT32×4	14	16
hub_rx_ts	30	4
hub_seq	34	2

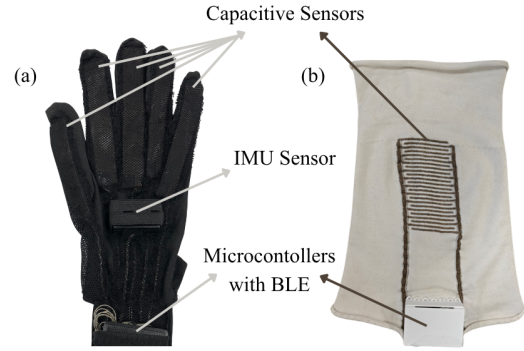


Fig. 2. Textile-based sensors: (a) sensing glove, (b) knee brace.

the connection parameters on every server link, connects or disconnects servers to match the count, and forwards the new rate to each server as a control command, staggered across links to avoid exhausting the transmit-buffer pool. The same characteristics also carry a remote-reboot command, encoded as a reserved option in the rate field.

Each server applies the new rate and confirms via a GATT indication. Once every addressed server has confirmed or exhausted its retries, the hub returns a 3-byte control approval record, which includes a bitmask of failed servers and the count addressed, so the client learns exactly which servers approved the configuration and can verify a run before trusting it. A server that fails to confirm keeps its previous setting, resending the runtime configuration record re-applies it. Thus, the control path is confirmed end-to-end.

D. Reliability and Flow Control

The hub carries every server's traffic, so it exhausts its receive and transmit resources as connections accumulate. To relax the congestion, the hub uses enlarged pools: 24 Asynchronous Connection-oriented Logical transport Transmit

(ACL-TX) buffers, 24 ATtribute protocol Transmit (ATT-TX) buffers, and 28 Receive-event (RX-event) buffers, relative to a single-connection server's requirements. Both ends negotiate a 64-byte ATtribute protocol Maximum Transmission Unit (ATT MTU) with Data Length Extension (DLE) so the whole mirrored record travels in one link-layer packet per connection event, without which the record would split across events.

IV. EXPERIMENTS

A. System Use Case with Real T-IoT Devices

Assume that two custom T-IoT devices as in Fig. 2 are used, glove as an input device for hand gestures and knee brace for locomotion in VR or both for the medical rehabilitation data capturing. The glove is built on an ESP32-WROOM-DA and the knee brace is built on an ESP32-C3. Each textile sensor utilizes different data: the glove sends five integers from the fingers in addition to IMU data, and the knee brace only exposes one integer. The self-describing packet format is designed to handle different textile-based sensors.

Each sensor is an independent, self-registering firmware module. Adding one requires a single module file, a build-time toggle, and an enum value, with no change to the wire format, the hub, or the clients. From the registered modules, the server device builds its own schema and data records at runtime. The firmware is C++ on Zephyr RTOS, with no dynamic memory allocation and no exceptions. All GATT services are registered at runtime, so a server device's sensor set can vary without recompiling the framework.

B. Experimental Framework

To characterize the transport itself rather than sensor heterogeneity, every server runs in a baseline mode during the measurements. Its real sensors are disabled and it streams a fixed-size, 50-byte payload, identical across all servers, at the notification rate configured. The T-IoT devices described previously demonstrate integration but are not involved in the sweep. All combinations of connection interval 20 ms, 50 ms, 75 ms and 100 ms, notification rate 10 ms, 25 ms, 50 ms and 100 ms, and server count 2, 4, 6, 8 and 10 were swept, with each repeated five times. Results were reported as *mean ± standard deviation* in terms of RTT and packet loss. For every configuration, the first 30 s is discarded and then 60 s is recorded. The hub is a Seeed XIAO BLE (nRF52840); server devices are a mix of nRF52840, STM32WB55, and ESP32-S3. The client is a laptop running Linux with the Python Bleak library. The hub-to-client link CI is 50 ms and fixed throughout the experiments.

C. Evaluation Criteria

Packet loss is measured on the hub-to-client link from gaps in the hub's forwarding sequence (*hub_seq*):

$$loss = \frac{g}{g + r}, \quad (1)$$

where g is the summed sequence gaps and r the received count. A gap is a record the hub forwarded but the client

never received. This global counter is purposely used instead of the per-server *seq*. It is robust to the per-server sequence resets that occur when a congested link disconnects and reconnects, which corrupt naive per-server counting under high loss. Because server-to-hub loss is negligible according to our experiments, this counter also captures essentially all end-to-end loss while isolating the bottleneck link. A configuration is included only if every server confirmed the runtime configuration record. A server that confirms yet delivers no packets during the recording is reported as a blacked-out server. To evaluate latency performance, RTT is measured using an 8-byte ping that the client writes to the hub and the hub echoes back.

V. RESULTS

A. Evaluation with Synthetic Data Transmission

Each heatmap in Fig. 3 and Fig. 4 show two of the three swept parameters. The missing parameters are fixed as 25 ms notification rate, 50 ms CI, and server count 6.

1) *Results on Packet Loss*: The packet loss heatmaps are shown in Fig. 3. The heatmap in Fig. 3 (a) shows the effect of active servers and notification rate. With low server counts and high notification rates, the offered load, which is *server count × notification rate*, is low, and the network manages it with low loss regardless of the CI. However, as the offered load increases, the loss gradually increases as well. Packet loss grows from 0 % to 10 % at 100 ms notification rate to 0 % to 80 % at 10 ms notification rate as server count increases from 2 to 10, with CI fixed at 50 ms. Fig. 3 (b) shows the effect of server-hub link CI and notification rate. It is evident that with high notification rates and low CI, the loss increases. It rises from 0 % at 100 ms CI to 10 % to 80 % at 20 ms CI as the notification rate decreases from 100 ms to 10 ms, with server count fixed at 6. Finally, Fig. 3 (c) shows the effect of server-hub link CI and server count, with notification rate fixed at 25 ms. As the server count grows and the CI shortens, the loss increases. It climbs from 0 % to 25 % at 100 ms CI to 0 % to 87 % at 20 ms CI as server count increases from 2 to 10.

2) *Results on Latency*: The RTT heatmaps in Fig. 4 follow a similar pattern to the packet loss. The surviving packets are delayed as the server-hub link CI shortens, server count grows, and notification rate increases. However, it is important to note that ping responses also drop in high-contention configurations; consequently, fewer RTT samples are available in those cells, so the reported mean may understate the true latency under the heaviest loss. The heatmap in Fig. 4 (a) shows the effect of active servers and notification rate, with CI fixed at 50 ms. Mean RTT climbs from 39 ms to 122 ms at 100 ms notification rate to 28 ms to 349 ms at 10 ms notification rate as the server count increases from 2 to 10. Fig. 3 (b) shows the effect of server-hub link CI and notification rate, with server count fixed at 6. It grows from 35 ms to 53 ms at 100 ms CI to 311 ms to 464 ms at 20 ms CI as notification rate decreases from 100 ms to 10 ms. The final heatmap Fig. 3 (c) shows the

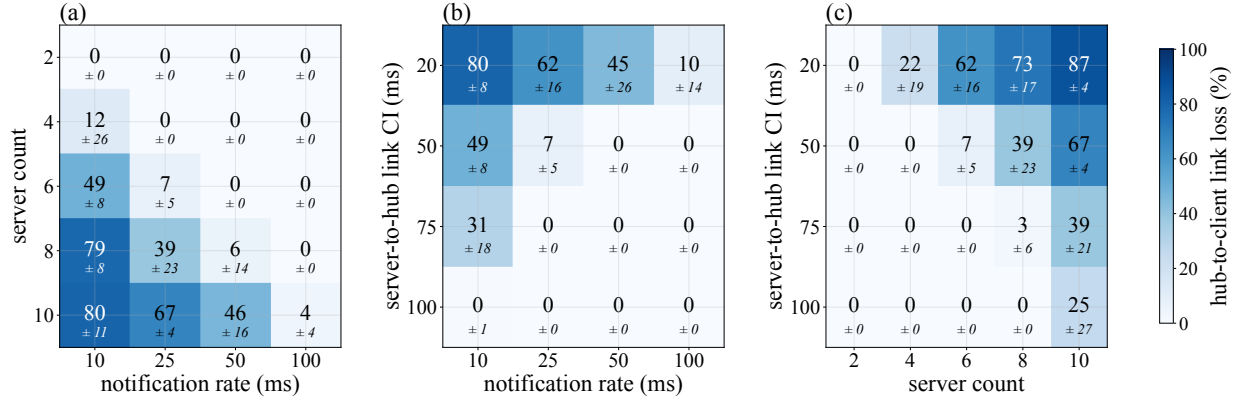


Fig. 3. Packet loss heatmaps across three parameters: (a) server count vs. notification rate, (b) server-to-hub link CI vs. notification rate, (c) server-to-hub link CI vs. server count.

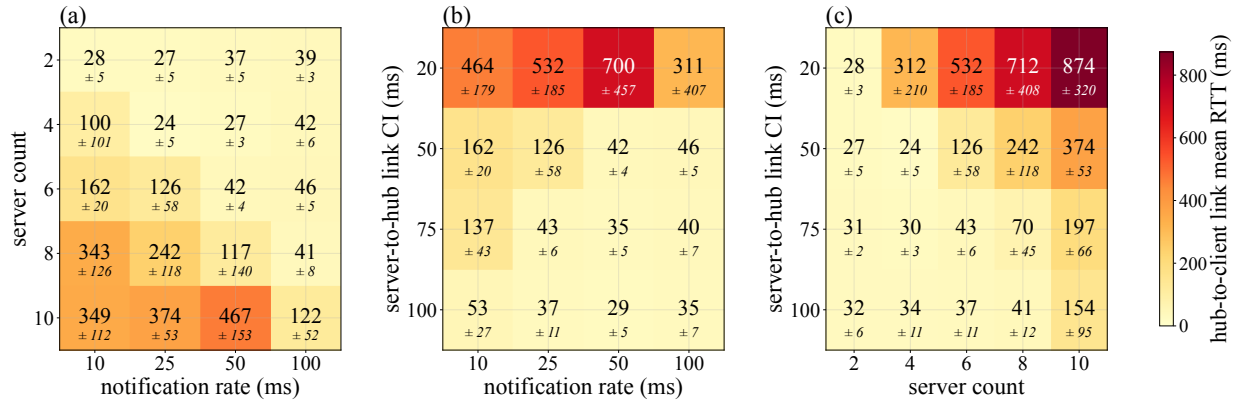


Fig. 4. RTT heatmaps across three parameters: (a) server count vs. notification rate, (b) server-to-hub link CI vs. notification rate, (c) server-to-hub link CI vs. server count.

effect of server-hub link CI and server count, with notification rate fixed at 25 ms. It climbs from 32 ms to 154 ms at 100 ms CI to 28 ms to 874 ms at 20 ms CI as server count increases from 2 to 10.

3) *Results on Contention and Fairness:* Fig. 5 shows servers missing from the recording even though they had approved the configuration, the notification rate is fixed as 25 ms. The trend is that the lower the CI, the more servers go missing, suggesting a contention and fairness issue.

B. Evaluation for the Use Case with Real T-IoT Devices

Streaming the glove and knee brace as the two connected servers, the framework sustained 0% loss across all server-hub link CI and notification rate configurations. The textile-based strain and IMU sensors are sampled at 50 Hz, so their effective packet rate is bounded by this configured sampling rate rather than the notification rate, keeping this realistic load below the congestion regime of the baseline sweep. This validates that realistic scenarios, with varying notification rates and payload sizes, perform within acceptable limits.

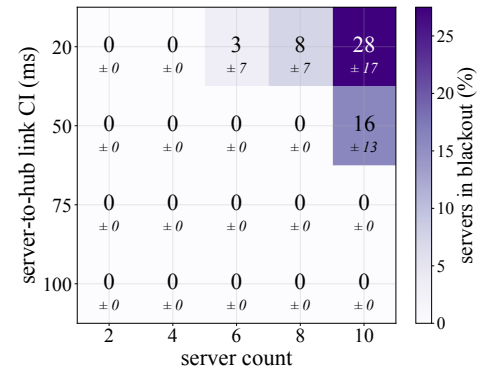


Fig. 5. The percentage of servers in blackout.

VI. DISCUSSION

In the architecture with the introduced framework given in Fig. 1, the main packet loss is caused by the single hub-to-client link. This link forces the hub to continuously drain its transmission buffers, since data comes from every single server and each notification must be forwarded to the client. To address this contention, the buffers of the hub were increased,

and different CI values were tested. However, because the packets are small and numerous, a bundling approach is necessary, especially for high-load setups. That said, if the setup does not require high server counts and high notification rates, as in the T-IoT scenario, the packet loss is below 1% and no bundling is required.

One aspect of the server-hub link CI is that with low CI values, the hub has less space to schedule the connections. This has a direct consequence as seen in Fig. 5: some servers are never allocated a time slot for the hub to forward their packets. These servers are missing from the recorded data; in other words, they black out. It is known that the connections are alive, since the servers need to approve the configurations, and partial approvals are not included in the results. Hence, higher server counts require higher CI for the hub to preserve fairness across connections.

Fig. 3 and Fig. 4 share similar trends. It is clear that, to keep packet loss and RTT below acceptable thresholds with higher server counts, CI must be high and the notification rate should be low. With server count below 6, most of the configurations demonstrate packet loss below 1% and RTT below 50 ms. When more servers are involved, the parameters should be tuned carefully. Unlike the single-peripheral star network of [16], our architecture adds a mirroring hub between the servers and the client. Even so, our finding that higher CI supports more simultaneous nodes without degradation is consistent with their result that maximum supportable network size increases with CI.

Some of the findings while carrying out the research are: vendor choice does not significantly affect server-device performance, though it is an important consideration for the hub, which must sustain at least 11 (10 servers plus 1 client) simultaneous connections to meet target WBAN applications as ascertained by nRF based devices.

As a limitation, the framework is evaluated on the benchtop rather than on the human body, since a baseline for comparison is necessary before the effects of body shadowing, posture, and movement can be measured. The connect-time verification is a lightweight security check rather than link-layer security, again to preserve a clean baseline for link performance.

VII. CONCLUSION

In this work, a runtime-reconfigurable BLE framework that implements a WBAN through a dual-role hub, exposing all server devices to the client over a single connection whose parameters can be reconfigured at runtime is introduced. The system is evaluated with packet loss and RTT across connection interval, server count, and notification rate on a heterogeneous mix of three hardware platforms. The key finding is that both loss and RTT degrade under high load. However, at realistic textile-sensor rates and server count, the framework performs with near zero loss and RTT below 100 ms. By prioritizing real BLE measurements over simulation, this work addresses the gap in real-hardware, multi-node BLE implementations and provides experimental findings for future WBAN systems.

Future work includes on-body trials with additional T-IoT devices, BLE link security, packet bundling, automatic load balancing algorithms, and energy consumption measurements.

REFERENCES

- [1] V. J. Aski, V. S. Dhaka, A. Parashar, S. Kumar, and I. Rida, "Internet of Things in healthcare: A survey on protocol standards, enabling technologies, WBAN architectures and open issues," *Physical Communication*, vol. 60, art. 102103, 2023.
- [2] J. Y. Khan, M. R. Yuce, G. Bulger, and B. Harding, "Wireless body area network (WBAN) design techniques and performance evaluation," *Journal of Medical Systems*, vol. 36, no. 3, pp. 1441–1457, 2012.
- [3] K. Ozlem, C. Gumus, A. F. Yilmaz, A. Tuncay Atalay, O. Atalay, and G. Ince, "Cloud-based control system with sensing and actuating textile-based IoT gloves for telerehabilitation applications," *Advanced Intelligent Systems*, vol. 7, no. 8, art. 2400894, 2025.
- [4] A. F. Yilmaz, I. Ahmed, C. Gumus, K. Ozlem, *et al.*, "Highly stretchable textile knitted interdigital sensor for wearable technology applications," *Advanced Sensor Research*, vol. 3, no. 2, art. 2300121, 2024.
- [5] K. Ozlem, A. Tuncay Atalay, O. Atalay, and G. Ince, "FogETex: Fog computing framework for electronic textile applications," *IEEE Internet of Things Journal*, vol. 12, no. 6, pp. 6856–6874, 2024.
- [6] Q. Liu, K. G. Mkongwa, and C. Zhang, "Performance issues in wireless body area networks for the healthcare application: a survey and future prospects," *SN Applied Sciences*, vol. 3, no. 2, art. 155, 2021.
- [7] I. Al-Barazanchi, W. Hashim, A. A. Alkahtani, H. H. Abbas, and H. R. Abdulshaheed, "Overview of WBAN from literature survey to application implementation," in *Proc. 8th Int. Conf. Electrical Engineering, Computer Science and Informatics (EECSI)*, 2021.
- [8] A. Nadeem, M. A. Hussain, O. Owais, A. Salam, S. Iqbal, and K. Ahsan, "Application specific study, analysis and classification of body area wireless sensor network applications," *Computer Networks*, vol. 83, pp. 363–380, 2015.
- [9] G. Koulouras, S. Katsoulis, and F. Zantalis, "Evolution of Bluetooth technology: BLE in the IoT ecosystem," *Sensors*, vol. 25, no. 4, art. 996, 2025.
- [10] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of Bluetooth Low Energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11734–11753, 2012.
- [11] M. Hosseinzadeh, A. Haider, S. Ali, M. Mohammadi, A. Mehranzadeh, T. Pornaveetus, and J. Lansky, "Enhancing DBSCAN clustering with fuzzy system to improve IoT-based WBAN performance," *Scientific Reports*, vol. 15, no. 1, art. 28404, 2025.
- [12] *IEEE Standard for Local and Metropolitan Area Networks—Part 15.6: Wireless Body Area Networks*, IEEE Std 802.15.6-2012, Feb. 2012.
- [13] J. Tosi, F. Taffoni, M. Santacatterina, R. Sannino, and D. Formica, "Performance evaluation of Bluetooth Low Energy: A systematic review," *Sensors*, vol. 17, no. 12, art. 2898, 2017.
- [14] M. Gupta, S. Tanwar, A. Rana, and H. Walia, "Smart healthcare monitoring system using wireless body area network," in *Proc. 9th Int. Conf. Reliability, Infocom Technologies and Optimization (ICRITO)*, 2021.
- [15] V. V. Tipparaju, K. R. Mallires, D. Wang, F. Tsow, and X. Xian, "Mitigation of data packet loss in Bluetooth Low Energy-based wearable healthcare ecosystem," *Biosensors*, vol. 11, no. 10, art. 350, 2021.
- [16] K. J. Rajotte, A. Wooding, B. E. McDonald, *et al.*, "Evaluation of a BLE star network for a wireless wearable prosthesis/orthosis controller," *Applied Sciences*, vol. 14, no. 22, art. 10455, 2024.
- [17] N. Landra, D. Demarchi, and P. M. Ros, "SharkTooth: A scalable real-time algorithm for BLE-based wireless body sensor networks synchronization," *IEEE Internet of Things Journal*, vol. 12, no. 22, pp. 46174–46192, 2025.
- [18] N. Krull, L. Schulthess, M. Magno, L. Benini, and C. Leitner, "Wireless low-latency synchronization for body-worn multi-node systems in sports," in *Proc. 2025 IEEE 21st Int. Conf. Body Sensor Networks (BSN)*, 2025, pp. 1–4.
- [19] Z. Zhou, C. Shen, T. Cao, and H. W. Huang, "Enhanced-BLE: A hybrid BLE-ESB framework for dynamically reconfigurable and energy-efficient 2.4 GHz IoT communication," arXiv preprint arXiv:2605.21270, 2026.