Distributed Automated Testing Framework for Bluetooth Mesh Applications

Jorg Wieme
imec - IDLab
Ghent University
Ghent, Belgium
jorg.wieme@ugent.be

Mathias Baert
imec - IDLab
Ghent University
Ghent, Belgium
mathias.baert@ugent.be

Jeroen Hoebeke
imec - IDLab
Ghent University
Ghent, Belgium
jeroen.hoebeke@ugent.be

***Abstract*—We introduce a comprehensive testing framework built on top of a Bluetooth Mesh Testbed, showcasing a blend of user-friendly design, swift implementation, and adaptability to diverse testing scenarios and technologies. The framework's modular architecture ensures syntactic validation, precise event scheduling, and efficient execution. Its well-defined syntax facilitates easy updates, catering to evolving requirements and maintaining agility. Operating independently across multiple nodes and accommodating various events, whether repetitive or one-time, the framework proves to be a versatile and robust solution for conducting tests. Tools like shared disks and synchronized clocks, provided by the Testbed, further enhance the framework's capability to schedule network flow behavior and gain insights into network performance. Overall, our testing framework presents a valuable tool for researchers and practitioners seeking a seamless and flexible approach to evaluate Bluetooth Mesh networks in a realistic environment.**

***Index Terms*—Bluetooth Mesh, Automated testing, Distributed testing, Testbed, Reproducibility**

I. Introduction

In recent years, Bluetooth Low Energy (BLE) has established a significant presence in the consumer device market, representing the low-power variant of Bluetooth Classic [1]. The technology is aimed at single-hop communication between two entities. The concept of setting up a multi-hop communication network that supports a variety of Internet of Things (IoT) applications on top, has gained more interest in recent years. In response to this evolving landscape, Bluetooth SIG introduced Bluetooth Mesh [2]. It involves establishing a topology over the existing BLE advertising layer and disseminating messages through a flooding-based approach.

Like any communication technology, Bluetooth Mesh undergoes frequent research aimed at enhancing its capabilities and addressing technological challenges. This is evident in various focused studies, such as relay selection [3], reliability [4], and other challenges [5]. Despite the diversity in research topics, all these investigations utilize tests to validate their findings and draw conclusions. Some of these tests are simulated, while others are performed in live Testbeds.

In our previous works [6] [7], we suggest overseeing and controlling a Bluetooth Mesh network through the implementation of a Digital Twin Network (DTN). DTNs are a virtual representation of a physical network, utilizing theoretical models and simulations to dynamically manage and optimize network configurations. To validate the effectiveness of the proposed solutions, conducting tests in a genuine environment was crucial in order to be exposed to real and unpredictable interference from individuals. To achieve this, a Testbed was employed. In our initial work [6], every aspect of the test was handled manually, from the configuration of each individual node to the transmission of specific messages. As anticipated, this manual approach proved to be neither scalable in terms of duration and traffic volume nor conducive to simultaneous testing. Therefore, our work in [7], which expands upon the ongoing DTN research related to Bluetooth Mesh, employed a more automated testing framework to perform the evaluation campaign. However, a comprehensive explanation of the testing procedures and the design of the testing framework itself, has not been provided in previous works. Consequently, this paper fills this gap by introducing a testing framework that aims to facilitate distributed testing with minimal manual interaction. The implementation of the testing framework is provided as part of the paper and can be found here [8]. Additionally, a binary version of the Nordic SDK is available upon request by contacting the first author. In summary, the contributions and key points of this framework can be outlined as follows:

- Reproducibility, where possible, is prioritized.
- Concurrent execution of multiple scenarios mirrors real-life situations.
- Easily definable tests without requiring knowledge of network and Testbed details.
- Internal safety checks alert users for potential issues.
- Tests operate autonomously, but manual adjustments can be made.
- User-friendliness enabled by an efficient and straightforward implementation.
- Can serve as benchmarking tool to compare diverse network configurations.
- Tailored for Bluetooth Mesh but applicable to any network communication.

The subsequent segments of this paper are organized as follows: Section II reviews prior research on Testbeds and frameworks. This is followed by Section III, which details the assumptions and fundamental requirements guiding the

Testbed. Section IV explores the framework and its functionalities. Lastly, Section V serves as a brief conclusion.

II. Related Work

Considerable research has been undertaken regarding Testbeds. The authors in [9] introduce an automated experimentation framework for engineering distributed systems, using models to characterize system components, Testbeds, and client behaviors in testing of specific scenarios. The work in [10] highlights the growing significance of networking Testbeds in shaping communication technologies. Towards addressing the challenges of conducting tests in federated, distributed, and heterogeneous Testbeds, the work in [11] introduces a flexible and lightweight framework that facilitates the management of Testbeds.

Moreover, equivalent research has been carried out for Wireless Sensor Networks (WSNs). The authors in [12] present a WSN testing framework, combining test-driven development, unit testing, and remote integration testing for tailored application testing. Additionally, the work in [13] provides an automatic, effective, and reliable testing mechanism for wireless devices during development and service verification before deployments. Pooling based frameworks such as the one presented in [14], act as distributed performance-testing frameworks that can automate service evaluation.

Research on Wireless Mesh Network (WMN) has concentrated on performance evaluation, as seen in [15], utilizing Testbeds. Furthermore, the work in [16] undertakes the evaluation of Bluetooth Mesh applications to a similar degree. However, their manual testing approach, which is guided by state diagrams, is specifically tailored for single tests, whereas our emphasis is on a comprehensive testing framework.

III. Testbed Setup

As mentioned earlier, the utilization of a Testbed is crucial to be able to conduct tests in a genuine environment. Therefore, this section outlines the selected Testbed and its specifications. Additionally, a set of necessary and highly advantageous assumptions related to the Testbed are provided to enhance the speed and usability of the framework.

A. Used Infrastructure

Several testbeds, including FIT IoT-LAB [17], TWIST [18], and others, offer potential for experimenting with the proposed framework. This study will concentrate on the IDLab OfficeLab [19], as we are a part of the research group that developed this testbed. IDLab OfficeLab features an environment with 110 nodes emulating an office space across multiple floors. Each floor is designed to replicate an authentic office setting, where genuine interference is introduced through, among others, active Wi-Fi usage and the presence of numerous BLE devices, either for personal or academic purposes. It should be noted that the operation of the framework remains unaffected by the number of nodes. This Testbed facilitates experimentation with various technologies including BLE, Wi-Fi, UWB, and others.

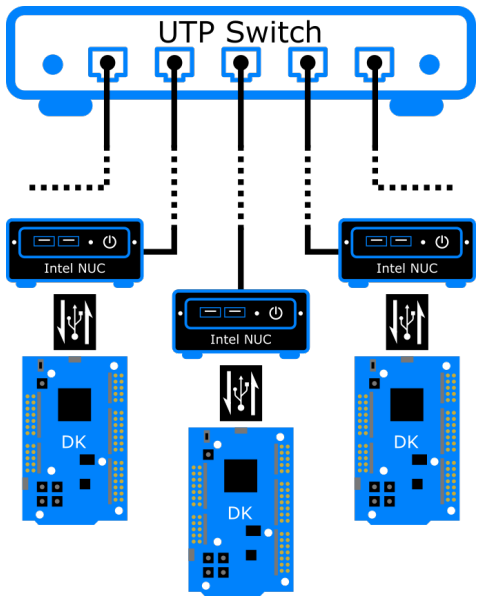

Fig. 1. Graphic representation showcasing how nodes are interconnected within the testbed via UTP.

In this Testbed, a logical node comprises an Intel NUC D54250WYKH equipped with an Intel Core i5-4250U processor and 8GB of RAM, running Ubuntu 18.04.2 LTS. These NUCs are interconnected via UTP and share the same network, with individual remote accessibility through the cloud. Additionally, each NUC, as depicted in Figure 1, is connected via USB to either a Nordic Semiconductor nRF52832 or nRF52840 Development Kit (DK). These DKs serve as the actual devices participating in the Bluetooth Mesh network. The Nordic devices themselves employ a customized version of the Nordic Semiconductor Mesh SDK [20]. The modified SDK was developed as part of the work in [7] and it facilitates communication with the NUCs through the UART protocol.

B. Testbed Requirements

As mentioned in the preceding paragraph, all the nodes in the employed Testbed are situated in the same network and the entire Testbed is accessible through the cloud. It is not mandatory for it to be accessible through the cloud if security requirements dictate otherwise. Although this may pose an inconvenience for swiftly deploying tests without being physically connected, it does not impede the framework's functionality. It is essential, however, that the nodes are within the same network, or at least every participating node can communicate with a central unit. This aspect will be further explained in the following paragraphs.

Being within the same network easily provides the advantage of having a shared disk, as is the case with the employed network. In this configuration, every individual NUC has access to this shared disk and some local storage. The primary benefit is the avoidance of dealing with multiple copies of the same file, encompassing both the framework code and the files defining a potential test. Another advantage lies in the central-

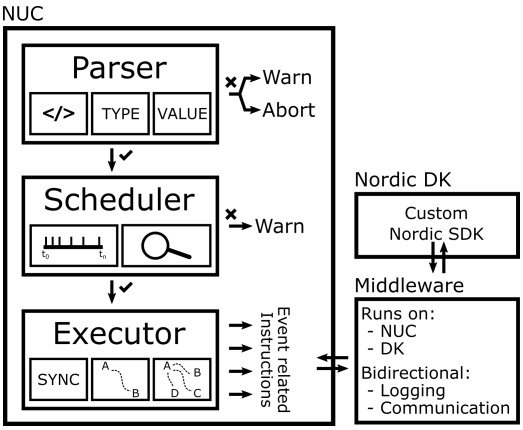

Fig. 2. Schematic representation of the modular structured framework.

ization of log file storage, accessible by any entity connected to this shared disk. This arrangement enables the construction of a result analyzing tool on top of the framework-generated results without the need for additional file copies, as this is facilitated by the shared disk implementation. It is important to note that a shared disk is not mandatory, but once again enhances the ease and speed of use.

To ensure the proper operation of the framework, an assumption is made that all participating NUCs in the network are synchronized. In the selected Testbed, this synchronization is achieved using Chrony [21]. Without this synchronization, certain assumptions made by the framework, such as time-sensitive scheduling of multiple events, cannot be guaranteed. Additionally, the Nordic DK does not possess an absolute clock but rather a relative one that initiates when the devices are powered, and this clock tends to drift relatively quickly. Although this does not directly impact the framework, it may affect the subsequent analysis. However, the framework addresses this issue, as explained in Section IV.

IV. Framework

Due to the features and advantages derived from the utilization of the testbed [7], the actual framework can be perceived as lightweight and easily implementable. The incorporation of previously mentioned features enables the framework to operate on each node individually, without the need to monitor the connections among all nodes. The framework itself is coded in Python, the design will be elucidated in the subsequent paragraphs. Following this, the syntax for test definition is described, contributing to the framework's ease of use and facilitating the reproduction of tests. To conclude this section, we also provide a demonstration of how the framework is utilized.

A. Framework Description

The framework design comprises three modules, as illustrated in Figure 2. The first module serves as a parser for interpreting test files written in JSON format. This choice ensures readability for both the test author, other users and the framework itself. The framework undergoes a validation process for the provided test, to find out whether it adheres to the predefined syntax. Furthermore, certain fields allow the definition of ranges for expected values or strings. In the event of any parser check failures, the severity determines whether the framework notifies the user or ceases execution altogether.

After successful validation of the syntax by the parser with no failures, it proceeds to forward the internal structure to the scheduler. The scheduler, operating uniquely on each node, manages the scheduling of all events specified for that particular node. During this scheduling process, the scheduler validates whether the requested test may lead to any collisions and notifies users so that they can account for this potential concern when verifying results. Once the schedule is generated and checked, it gets forwarded to the final module. An example of a collision could be attempting to send two events at exactly the same time.

The final module is the executor and takes charge of executing the events scheduled by the scheduler, employing a multi-threaded approach where each thread serves a distinct purpose. One primary thread oversees the overall test and concludes when the timer expires, signaling all other events to cease as well. Additionally, individual threads are instantiated for each defined event, running at the specified intervals. These events encompass a broad spectrum of actions, tailored to the underlying tested technology and hardware. For instance, in the context of Bluetooth Mesh, these actions could involve tasks such as sending a message to a designated destination, broadcasting a multicast message, or delving into more research-specific tasks, like adjusting internal settings such as enabling or disabling its relaying feature. It's noteworthy that these events and operations could operate repetitively or only once, depending on the nature of the test. Furthermore, owing to the multi-threading architecture, users retain the flexibility to manually perform events on the command line of the framework if desired.

The framework is not responsible for determining which data to collect; this is contingent on the tester's preferences and is connected to the underlying technology. For instance, if measuring queue filling is desired, the tester is responsible for implementing appropriate logging operations. These logging operations are implemented alongside the correlated operations requested by the framework. In our case, a debugger was attached to collect this generated logging information and forward it to a file on the shared disk.

Lastly, the time precision of the framework is limited by the underlying hardware devices. Since UART is utilized to communicate with these devices, the time precision remains indeterminate, as occasional communication losses can occur. However, to address this issue, the framework includes a configurable parameter to set a timeout in case communication is lost, thereby facilitating the restart of the current executor event.

B. Defining Tests

The expected test syntax is provided in Listing 1. The syntax for test definition is crafted to allow seamless updates during the exploration of new research focus points.

Initially, we encounter the global fields, universally interpreted by all NUCs, ensuring adherence to shared requests and rules. An integral field, particularly relevant when working with Nordic DKs, is the *sync_timer*. Given the relative nature of the clock in Nordic DKs, which tends to drift rapidly, the clock of the Nordic Devices is synchronized to the clock of the NUC (over UART), every *sync_timer* interval, to prevent the generation of incoherent data results. The *title* and *description* options are designed for quick comprehension of the specific test, also visible in the logging process for retrospective identification. As the framework operates distributively across multiple nodes without mutual awareness, defining a *random_seed* becomes necessary for reproducibility, especially since the scheduler incorporates randomness into recurring events. While not mandatory, it is recommended for significant tests. Lastly, *nodes* encompass a list of all nodes intended to participate in the events.

Listing 1. Fundamental JSON syntax and structure for defining tests.

```
{
    "sync_timer": int,
    "title": str,
    "description": str,
    "duration": int,
    "random_seed": int,
    "nodes": [
        {
            "mac": "00:01:02:03:04:05",
            "events": [
                {
                    "id": int,
                    "type": str,
                    "start_time": int,
                    "start_random": int,
                    "interval_time": int,
                    "interval_random": int,
                    "event_specific_1": ...,
                    "event_specific_2": ...
                }, { ... }
            ]
        }, { "mac": "01:02:03:04:05:06", ... }
    ]
}
```

For node-specific instructions, unique identifiers are employed, and in the context of Bluetooth Mesh, MAC addresses are utilized. However, any identifier could suffice. The only additional field in this section is the *events* field, encompassing the events designated for that specific node. Upon initiating the framework and successfully passing the syntax screening, each node searches for its respective MAC address and commences scheduling the corresponding events. In the current iteration, knowing the MAC addresses of the devices is required for participation. However, as a potential feature, the framework can be easily adapted with minimal additional implementation to support groups, thus removing the sole restriction of requiring MAC addresses. Each event instruction has a distinctive *id*. Following that is the *type* field, indicating the specific predefined event it references. Some events necessitate additional fields, while others do not, undergoing verification by the syntax parser. For example, a specific field can indicate an enforced transmit power or transmission channel. However, every event shares the *start_time* and *start_random* attributes, indicating the initial occurrence time of the event and, if non-zero, specifying the random boundary from which to select. A similar scenario applies to *interval_time* and *interval_random*, determining the duration between consecutive occurrences. It's worth noting that if *interval_time* is set to zero, the event does not repeat at all.

The work in [7] employed the described framework and associated JSON structure, to perform an evaluation campaign in a Bluetooth Mesh network. One of these tests involved the repetition of sensor traffic at a specified frequency, complemented by emulated light traffic achieved through the transmission of light switch information with a random factor. Throughout the process, extensive data logging on the NUCs was conducted for subsequent analysis.

C. Framework Usage

The initial step in utilizing the framework involves determining the type of experiment to be conducted. The user translates the envisioned experiment to a file, following the previously described syntax, and uploads it to the nodes for access. In our case, this file was placed on the shared drive. Next, a remote access point is instantiated for each participating node. The setup begins with configuring the network (i.e. supported application flows per node) and connecting the debugger for logging purposes. Finally, the framework code is initiated on each node to process the experiment file. Next to this, the framework generates information for each event to enable user tracking. The user is notified upon completion of the experiment and informed of potential issues that occurred during execution. Two possible extensions are identified: enabling scheduling of multiple occurrences of an event in absolute time (instead of interval-based) and adding the configuration of supported application flows per node, to the experiment file. The latter is currently pre-configured beforehand.

V. Conclusions

The proposed testing framework, built on top of a robust Testbed, combines user-friendliness, rapid implementation, and adaptability to diverse testing scenarios. Its modular structure ensures precise syntactic validation, event scheduling, and efficient execution. The well-defined syntax allows easy updates for evolving requirements. The framework's capacity to operate independently across multiple nodes, complemented by manual command-line interactions, adds an extra layer of flexibility. Furthermore, its adaptability to handle various events, whether repetitive or one-time, highlights its versatility as a robust solution for conducting comprehensive and flexible tests within wireless networks.

References

[1] Bluetooth SIG, "Bluetooth specification version 4.0," https://www.bluetooth.com/specifications/specs/core-specification-4-0/, 2010.

[2] ——, "Mesh profile specification: 1.0.1," https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/, 2019.

[3] E. A. J. Hansen, M. H. Nielsen, D. E. Serup, R. J. Williams, T. K. Madsen, and R. Abildgren, "On relay selection approaches in bluetooth mesh networks," in *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2018, pp. 1–5.

[4] R. Rondón, A. Mahmood, S. Grimaldi, and M. Gidlund, "Understanding the performance of bluetooth mesh: Reliability, delay, and scalability analysis," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2089–2101, 2020.

[5] A. Hernández-Solana, D. Pérez-Díaz-De-Cerio, M. García-Lozano, A. V. Bardají, and J.-L. Valenzuela, "Bluetooth mesh analysis, issues, and challenges," *IEEE Access*, vol. 8, pp. 53 784–53 800, 2020.

[6] M. Baert, E. De Poorter, and J. Hoebeke, "A digital communication twin for performance prediction and management of bluetooth mesh networks," in *Proceedings of the 17th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, ser. Q2SWinet '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1–10. [Online]. Available: https://doi.org/10.1145/3479242.3487327

[7] J. Wieme, M. Baert, and J. Hoebeke, "Managing a qos-enabled bluetooth mesh network using a digital twin network: An experimental evaluation," *Internet of Things*, vol. 25, p. 101023, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660523003463

[8] J. Wieme, "Distributed testing framework," https://gitlab.ilabt.imec.be/jowieme/distributedtestingframework.

[9] Y. Wang, M. J. Rutherford, A. Carzaniga, and A. L. Wolf, "Automating experimentation on distributed testbeds," in *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 164–173. [Online]. Available: https://doi.org/10.1145/1101908.1101934

[10] T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar, "Omf: A control and management framework for networking testbeds," *SIGOPS Oper. Syst. Rev.*, vol. 43, no. 4, p. 54–59, jan 2010. [Online]. Available: https://doi.org/10.1145/1713254.1713267

[11] M. Wachs, N. Herold, S.-A. Posselt, F. Dold, and G. Carle, "Gplmt: A lightweight experimentation and testbed management framework," in *Passive and Active Measurement*, T. Karagiannis and X. Dimitropoulos, Eds. Cham: Springer International Publishing, 2016, pp. 165–176.

[12] O. Banias and D. I. Curiac, "Wireless sensor network software testing framework," in *2010 International Joint Conference on Computational Cybernetics and Technical Informatics*, 2010, pp. 517–521.

[13] l. liu, w. wei, and j. li, "Wireless communication system automation testing framework," in *2007 International Conference on Wireless Communications, Networking and Mobile Computing*, 2007, pp. 2981–2984.

[14] C. Dumitrescu, I. Raicu, M. Ripeanu, and I. Foster, "Diperf: an automated distributed performance testing framework," in *Fifth IEEE/ACM International Workshop on Grid Computing*, 2004, pp. 289–296.

[15] N. Abu Ali, E. Ekram, A. Eljasmy, and K. Shuaib, "Measured delay distribution in a wireless mesh network test-bed," in *2008 IEEE/ACS International Conference on Computer Systems and Applications*, 2008, pp. 236–240.

[16] L. Eras, F. Domínguez, and C. Martinez, "Viability characterization of a proof-of-concept bluetooth mesh smart building application," *International Journal of Distributed Sensor Networks*, vol. 18, no. 5, p. 15501329221097819, 2022. [Online]. Available: https://doi.org/10.1177/15501329221097819

[17] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, "Fit iot-lab: A large scale open experimental iot testbed," Milan, Italy, Dec. 2015. [Online]. Available: https://hal.inria.fr/hal-01213938

[18] V. Handziski, A. Köpke, A. Willig, and A. Wolisz, "Twist: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks," in *Proceedings of the 2nd International Workshop on Multi-Hop Ad Hoc Networks: From Theory to Reality*, ser. REALMAN '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 63–70. [Online]. Available: https://doi.org/10.1145/1132983.1132995

[19] imec IDLab UGent, "Testbed officelab," https://boss.wilab1.ilabt.iminds.be/inventory/?viewMode=inventory, 2024.

[20] Nordic Semiconductor, "nrf5 sdk for mesh," https://www.nordicsemi.com/Products/Development-software/nrf5-sdk-for-mesh, 2024.

[21] R. Curnow and M. Lichvar, "Chrony," https://chrony-project.org, 2024.