Towards Efficient BLE Mesh: Design of an Autonomous Network Joining Algorithm

Clara Nieto-Taladriz, Yuri Murillo and Sofie Pollin
Department of Electrical Engineering, University of Leuven, Leuven, 3001, Belgium
clara.nietotaladrizmoreno@student.kuleuven.be, yuri.murillo@esat.kuleuven.be

***Abstract*—The Internet of Things (IoT) opens the doors to a digital revolution, but requires a robust protocol to wirelessly interconnect a large number of devices. Bluetooth Low Energy (BLE) Mesh emerges as a suitable candidate, solving the range limitations of original BLE. Nevertheless, the main limitation is shifted to the scatternet architecture: how the considerable number of end systems are interconnected to ensure the network efficiency and scalability. As of today, some timid solutions have emerged, though the most relevant parameters for the best parent selection in BLE mesh network joining procedures have not been identified yet. In this work, we perform a thorough exploration of different device and environmental parameters in a BLE mesh testbed to analyze their impact in the overall network figures of merit, such as end-to-end delay and packet delivery ratio (PDR). According to the inferred relevance, the second part of the work implements and measures the proposed parent selection algorithm. The implementation is based on the open source Fruitymesh protocol and is used as a baseline. The results reflect an enhancement in the network scalability and fairness, accomplishing a delay and PDR improvement of 26% and 10% respectively, and the avoidance of saturated branches of 24%.**

***Index Terms*—BLE, BLE mesh, WSN, IoT, multi-hop communication**

I. Introduction

The integration of wireless networks in our society acquires promising forecasts, powered by the IoT emergence [1]. However, they require the interconnection of a massive number of devices. Bluetooth Low Energy (BLE) emerges forcefully for this purpose [2] thanks to its low cost and strong presence in the marketplace. Nonetheless, BLE is a single-hop technology with a coverage range of scarcely 10-15 meters in indoor applications [3]. The BLE mesh [4] release granted BLE with a mesh structure and multi-hop communication, so the coverage range can be easily augmented by enlarging the scatternet.

The BLE mesh relaying is flooding-based. Albeit is simple and straightforward, it suffers from high packet collision, unnecessary high power consumption and its set of rules does not allow fine-tuning. As a consequence, the drawback is shifted to the scatternet structure and how the different end systems are connected within the network to ensure the network efficiency.

Shifting BLE mesh towards a routing-based, connected network may solve this issue. Nonetheless, BLE nodes do not all share the same effectiveness in relaying messages nor holding new nodes, due to the heterogeneity germane to IoT applications. Therefore, the different routing-based alternatives also reinforce the relevance of the topology for the overall network performance.

A multi-hop protocol and the handling of messages with different priorities were already implemented in our BLE mesh testbed [5]. Hereby, this work proposes an algorithm to join the BLE network efficiently. Our contribution is twofold. First a wider and thoroughgoing analysis was performed to identify the most relevant parameters in the network joining process. Thereupon, they were weighted according to the inferred relevance and the resulting algorithm was measured and compared with the current procedure.

II. Current Framework

Currently, the BLE nodes use simultaneously two mesh protocols [5]: Trickle [6], a flooding scheme used for broadcasting advertisements; and FruityMesh [7], a connection-oriented protocol used for regular data to reduce the energy consumption. FruityMesh is a clustering protocol that tends to convergence into a large and common network, which emphasizes the relevance that the location of each end system acquires to guarantee the network performance. However, this issue is not considered with the current procedure (Fig. 1).

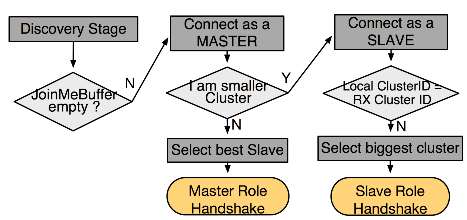


Fig. 1. Simplified flowchart of the FruityMesh network build up.

III. Proposed BLE mesh network joining framework

The proposed algorithm (Fig. 2) filters the neighbors suitable for a connection using the information extracted from broadcasted status messages and scores them. Afterwards, the highest-scored within the biggest cluster is chosen as a parent, and the $ACKfield$ of the joinMe packet acquires its identifier. This way, the node is able to explicitly specify its preferable connection.

The proposed scoring algorithm (Fig. 3) is compound by the slaves number (M), hops to the sink (H), buffer occupancy (B), connection interval (CI), link received signal strength (RSSI) (RL) and the RSSI between the neighbor and its master (RN).

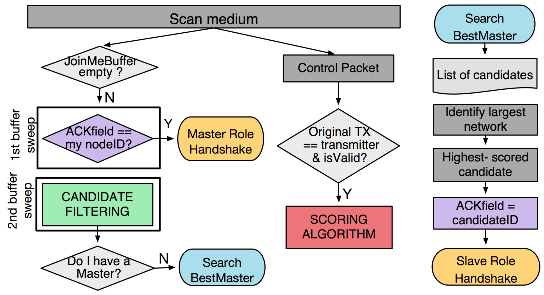


Fig. 2. Simplified flowchart of the proposed network build up.

```
(1)        if (RL or  RN) <= -82 :
                then remove candidate
(2)        if (B >= 60\% of B_max) :
                then remove candidate and children
(3)        else :
                S_x = 1500 - 21 H - 52 B -  26 CI - M
```

Fig. 3. Pseudocode of the proposed scoring algorithm.

The numerical values were chosen after a thorough analysis, measuring empirically the impact and relevance of each scoring parameter in the BLE mesh testbed performance. The latter is evaluated through two main figures of merit: the end-to-end delay and PDR. The chosen weights work as a ballpark.

IV. Experimental Results

The proposal was deployed in the training (11 BLE nodes) and in larger random-generated networks (16 nodes), using the original FruityMesh procedure as a baseline. First columns of Table I show the delay and PDR mean (μ) and deviation (σ) empirically measured in the new node. $\%Sat$ is the probability of the new device working in saturated branches, $\overline{N_{Hops}}$ the mean distance occupied by the latter and $avoid\_Sat$ denotes the avoided congested branches. The comparison in terms of delay, PDR and probability of saturation is depicted in Fig. 4.

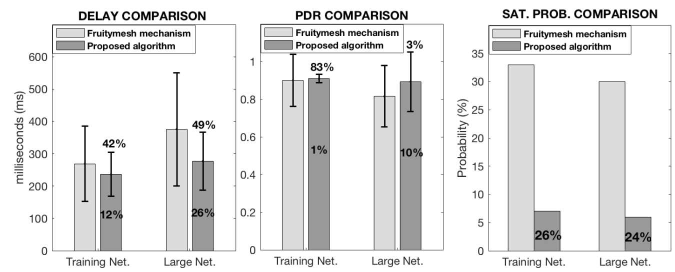
	μ_d	σ_d	μ_{PDR}	σ_{PDR}	% Sat	avoid_Sat	$\overline{N_{Hops}}$
TRAINING NETWORK	Applying the **Proposed** algorithm						
	236	68	0.91	0.02	7%	71%	2.4
	Applying the **Fruitymesh** mechanism						
	269	117	0.90	0.14	33%	-	2.6
LARGER NETWORKS	Applying the **Proposed** algorithm						
	277	90	0.89	0.158	6%	70%	2.7
	Applying the **Fruitymesh** mechanism						
	376	175	0.82	0.163	30%	-	3.5

TABLE I. Fruitymesh and proposed algorithm comparison.

In the training network both algorithms select the same $\overline{N_{Hops}}$. Since our algorithm considers the CI and chooses chiefly empty buffers, it attains a delay enhancement of 12% (Fig. 4(a)), which conspicuously improves in larger networks, reaching the 26%. The significant reduction of σ_d evinces a higher precision in choosing parents that guarantee low delays.

The PDR improvement accomplishes the 10% in larger networks (Fig. 4(b)) since the CI consideration allows selecting nodes with buffers that empty faster. Furthermore, the noteworthy σ_{PDR} reduction in small networks also denotes the tendency of choosing the best parent.

The proposed algorithm wisely discards bottlenecks. Moreover, discarding nodes without an empty buffer but selecting a child introduces the fairness concept. Notwithstanding, if



(a) Delay (b) PDR (c) Prob. of saturation

Fig. 4. Joining procedures comparison.

at the decision instant a device has a buffer occupancy fairly close to the threshold, choosing a child will fail on the saturation avoidance. This drawback justifies the meager σ_{PDR} reduction in larger scenarios. Nevertheless, it corresponds to an infrequent case. In Table I $avoid\_Sat$ indicates that the 70% of branches with high traffic load are successfully avoided.

Fig. 4(c) represents a reduction of 25% in the probability of operating in saturated environments. Since a link suffering from congestion flagrantly affects the whole branch, the improvement observed in the last two graphics in the new node can be extrapolated as an enhancement of the entire network.

V. Conclusions

This paper proposes a network building scheme for BLE mesh networks. Albeit in general enlarging the scatternet will always worsen the global performance, a good master and network position allows mitigating the adverse effects. The proposed algorithm allows integrating a new device under the principles of fairness and scalability. In the vast majority of cases, the new node joins the scatternet without jeopardizing the overall network performance and being capable of performing a befitting operation in terms of delay and PDR.

The scoring algorithm values work as a ballpark. Albeit they may differ from the optimum, they are accurate enough to improve the overall network performance figures of merit.

References

[1] Forbes, “2017 Roundup Of Internet Of Things Forecasts.” [Online]. Available: https://www.forbes.com/sites/louiscolumbus/2017/12/10/2017-roundup-of-internet-of-things-forecasts

[2] M. Collotta, G. Pau, T. Talty, and O. K. Tonguz, “Bluetooth 5: A concrete step forward toward the IoT,” *IEEE Communications Magazine*, vol. 56, no. 7, pp. 125–131, JULY 2018.

[3] P. D. Marco, R. Chirikov, P. Amin, and F. Militano, “Coverage analysis of Bluetooth low energy and IEEE 802.11ah for office scenario,” in *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Aug 2015, pp. 2283–2287.

[4] Bluetooth SIG, “BLE mesh profile specification 1.0,” July 2017. [Online]. Available: https://www.bluetooth.com/specifications/mesh-specifications

[5] Y. Murillo Mange, B. Reynders, A. Chiumento, and S. Pollin, “A Multiprotocol Low-Cost Automated Testbed for BLE Mesh,” *IEEE Communications Magazine*, 2018.

[6] P. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle: A Self-regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks,” in *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1*, ser. NSDI’04. Berkeley, CA, USA: USENIX Association, 2004, pp. 2–2.

[7] FruityMesh, “Fruitymesh,” accessed 22/08/2018. [Online]. Available: https://github.com/mwaylabs/fruitymesh/wiki