An all Wireless SDN Framework for BLE Mesh

YURI MURILLO, KU Leuven, Belgium

ALESSANDRO CHIUMENTO, KU Leuven, Belgium and Trinity College, Ireland

BRECHT REYNDERS, KU Leuven, Belgium

SOFIE POLLIN, KU Leuven, Belgium

The Internet of Things (IoT) paradigm combines the interconnection of massive amounts of battery constrained and low computational power devices with low latency and high reliability network requirements. Additionally, diverse end-to-end services and applications with different Quality of Service (QoS) requirements are expected to coexist in the same network infrastructure. Software Defined Networking (SDN) is a paradigm designed to solve these problems, but its implementation in wireless networks and specially in the resource constrained IoT systems is extremely challenging and has seen very limited adoption, since it requires isolation of data and control plane information flows and a reliable and scalable control plane. In this work, Bluetooth Low Energy (BLE) mesh is introduced as an adequate technology for an all wireless SDN-BLE implementation, which is a technology that has become the de-facto standard for IoT. The proposed SDN-BLE framework uses a routing network slice for the data plane information flow and a flooding network slice for the control plane information flow, ensuring their isolation while still being transmitted over the wireless medium. The design and implementation of all the classical SDN layers on a hybrid BLE mesh testbed is given, where the data plane is formed by the BLE nodes and the control plane can be centralized on a server or distributed over several WiFi gateways. Several controllers are described and implemented, allowing the framework to obtain end-to-end network knowledge in order to manage individual nodes over the air and adequate their behavior to meet application requirements. An experimental characterization of the SDN-BLE framework is given, where the impact of the different parameters of the system on the network reliability, overhead and energy consumption is studied. Additionally, the distributed versus centralized control plane operation modes are experimentally characterized, and it is shown that the distributed approach can provide the same performance as the centralized one when careful system design is performed. Finally, a proof of concept for the SDN-BLE framework is presented, where a network congestion is automatically detected and the nodes responsible of such congestion are identified and reconfigured over the air, bypassing the congested links, in order to resume regular network performance.

CCS Concepts: • **Networks** → **Network experimentation; Network performance analysis; Network measurement; Network management; Network design principles; Network protocol design; Network structure; Network dynamics; Network reliability; Network manageability.**

Additional Key Words and Phrases: SDN, IoT, WSN, BLE, BLE mesh, Testbed design, Trickle, FruityMesh.

ACM Reference Format:

Yuri Murillo, Alessandro Chiumento, Brecht Reynders, and Sofie Pollin. 2020. An all Wireless SDN Framework for BLE Mesh. *ACM Trans. Internet Things* 1, 1, Article 1 (January 2020), 30 pages. <https://doi.org/10.1145/3403581>

Authors' addresses: Yuri Murillo, yuri.murillo@esat.kuleuven.be, KU Leuven, Kasteelpark Arenberg 10, Leuven, Belgium, 3001; Alessandro Chiumento, KU Leuven, Leuven, Belgium, Trinity College, Dublin, Ireland, chiumena@tcd.ie; Brecht Reynders, brecht.reynders@esat.kuleuven.be, KU Leuven, Kasteelpark Arenberg 10, Leuven, Belgium, 3001; Sofie Pollin, sofie.pollin@esat.kuleuven.be, KU Leuven, Kasteelpark Arenberg 10, Leuven, Belgium, 3001.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Association for Computing Machinery.

2577-6207/2020/1-ART1 \$15.00

<https://doi.org/10.1145/3403581>

1 INTRODUCTION

1.1 SDN for IoT: Advantages

The Internet of Things (IoT) paradigm presents a massive interconnection of devices in dense networks that consist of low-power sensor nodes with reduced computational power and limited batteries. Therefore, both low energy expenditure and minimal data processing are necessary design goals, but these premises conflict with the additional high reliability and low latency requirements of IoT and Wireless Sensor Networks (WSN) [17]. These kind of networks also need to be able to run different applications with different end-to-end requirements and Quality of Service (QoS) necessities in the same infrastructure [17]. The aforementioned principles require reliable and robust networking protocols along with adaptive management solutions that scale with dynamic network performance in order to ensure low energy expenditure and efficient and scalable network management for delivering QoS at minimal energy cost.

Software Defined Networking (SDN) is a technological paradigm that enables easy and efficient network management and node reconfiguration with the aim of improving end-to-end performance and monitoring [8]. As networking evolved away from traditional static, sparse, single technology and single application architectures, it soon became evident that flexible, simple and efficient network provisioning and management solutions were needed [8]. SDN aims to provide them by 1) establishing a clear separation between data and control planes and 2) centralizing network intelligence into one entity [8]. In this context the data plane is formed by the actual nodes creating and forwarding the data packets, while the control plane consists of one or (less frequently) more intelligent controllers in charge of the network control and maintenance.

As a result of these two premises, implementing the SDN paradigm to WSN is advantageous for many reasons. The three main benefits highlighted in this work reside in terms of *energy efficiency*, *network management* (and specially its scalability) and *infrastructure reusability*.

First, the separation of the control and data plane allows to use *simpler and more energy efficient wireless nodes*, since their only task now becomes sensing the environment and forwarding packets. Other complex and power-intensive tasks such as network maintenance, data aggregation and information storage are shifted to the network controllers, which ideally are not as resource constrained as the nodes. Minimizing energy consumption across the network is then a key element in order to extend the life of WSN [2].

Second, the hierarchical organization and architectural idiosyncrasy of WSN are not universally defined, and may range from clearly vertically structured to pure horizontally peer-to-peer distributed, having static or dynamic nodes or even simultaneously including all of the above [2, 17]. Management of network topology, route negotiation and neighbor updating and discovery may then become tasks that are too computationally complex for the nodes and even potentially incur costly network overhead, especially when the network grows in size. The SDN architecture, by virtue of establishing a centralized network controller, can obtain a *global view of the end-to-end performance of the network* and real-time application requirements. Then, it may centrally design effective network policies and identify the effect of each node in the whole network, so individual node reconfiguration may be conducted to adequately satisfy global requirements more effectively.

Third, the centralized policing based on global knowledge and individual node addressing and reconfiguration *removes the application specific nature* of many WSN [17], allowing a single architecture to fit multiple purposes and applications while reusing the same network infrastructure.

1.2 SDN for IoT: Challenges

Continuing with the SDN discussion, the separation of data and control planes typically means that two kinds of network information flows are generated: one carrying data packets, which need

to be forwarded to the final destination; and one carrying control information, which informs the network controllers about the status of the nodes in order to facilitate network maintenance [18]. As a result, the two main requirements for SDN are [8, 17, 18]:

- (1) Ensuring **isolation of data and control plane information flows**.
- (2) Implementing a **reliable and scalable control plane**.

Isolation of the two flows is required so the data plane traffic is not compromised by the incorporation of this additional control plane traffic and vice-versa [18]. Reliability of the control plane is required in order to guarantee that both status updates from the nodes to the controllers and node reconfiguration from the controllers to the nodes arrive in a timely manner. However, centralizing the network intelligence presents drawbacks in terms of security and scalability, as it focuses the point of failure of the network into one entity [18].

SDN was originally designed for wired networks, where these requirements were easily achieved. Isolation of both flows was guaranteed by using two network interfaces and network slices; a reliable control plane was ensured since the wired infrastructure allows to provide reliable channels; and a scalable control plane and efficient while simple network controllers were achievable given the static nature of such networks [8]. When SDN is applied to wireless networks, such isolation is achieved by redirecting the control plane flow over a dedicated wired backbone, or alternatively over a secondary wireless interface using a dedicated technology that is able to guarantee reliable traffic delivery and allows to handle the increased energy demand and channel occupancy [17].

IoT and WSN generally have limited range, capacity, reliability and energy availability, and are composed by mobile nodes with single network interfaces and limited computational power, so directly applying SDN is usually not possible. Moreover, the wireless medium presents additional challenges that are particularly punishing in WSN given its aforementioned characteristics:

- (1) **Isolation** of data and control plane information flows is counteracted by its broadcast nature.
- (2) **Reliability** of control information is compromised, as the single network interface means that both flows will need to be implemented in an in-band scheme and will suffer from the same impediments in terms of interference, medium access and stochastic channel quality.

Depending on the targeted application constraints, design requirements and technology used, implementing an infrastructure that shares resources (channels, paths and/or timeslots) may be needed, while in other cases this isolation can be respected. Additionally, reliability and QoS may not be possible to achieve depending on the protocols used. In any case, the technology used to achieve the required SDN-WSN solution imposes additional challenges to the design and implementation process apart from the aforementioned ones.

1.3 SDN for IoT: BLE and BLE mesh as the SDN enabler

In this context, Bluetooth Low Energy (BLE) [3] has emerged as the de-facto standard for low power and low data rate WSN given its low cost and compatibility with a vast spectrum of off-the-shelf devices. Major adoption of BLE solutions in markets such as eHealth [13] or the smart home [7] has attracted both academic and industrial interest in order to analyze and optimize its performance in terms of throughput [10], delay [6] and energy consumption [33]. For this work a hospital scenario is considered, where the BLE nodes represent battery-powered sensors that monitor the patients' vital constants. Their indoor coverage typically range from 10 to 20 meters [3], but recent efforts have been carried out in order to introduce mesh capabilities to BLE, as more coverage can then be achieved simply by adding more low-cost BLE mesh devices that forward the packets to a final destination. Recent studies have shown that there are several factors to take into account when designing a BLE mesh solution that satisfies requirements on end-to-end delay, packet reliability, network overhead and energy consumption [27], where simultaneously meeting all of them is

not possible. As a result, the performance of BLE mesh technologies has been shown to vary significantly depending on the protocol implementation and design choices, such as the routing strategy [5]. This increased momentum led to the release of the first version of the official BLE mesh protocol in July 2017 [4], but this proposal represents a very simple initial step towards an optimal BLE mesh solution and research on characterization of its performance in real life applications and its optimization strategies is still sparse. Moreover, efficient network maintenance and node reconfiguration for BLE mesh is still an open question, and additionally all BLE mesh solutions in the literature still suffer from the aforementioned WSN impediments. This work elaborates on how all of these can be solved by a fully wireless SDN-BLE implementation, where its fully wireless nature means that these impediments can be solved without modifying the hardware in the nodes and ensuring that they can stay mobile, powered by their batteries and without being attached to a wired backbone, which is an important requirement for the application considered. In any case, to the best of the authors' knowledge no other work in the State of the Art (SoA) designs, implements and experimentally characterizes an SDN-BLE mesh solution, specifically addressing the isolation of data and control information flows and control plane reliability challenges.

1.4 Paper contributions and structure

The contribution of this paper is threefold:

- It motivates BLE mesh as a suitable SDN-WSN candidate and proposes a **SDN-BLE framework** with wireless isolation of data and control plane flows and reliable control plane. The control plane uses a flooding network slice using Trickle [19] and the data plane uses a routing network slice using FruityMesh [22] as their mesh protocols.
- It presents an implementation of the SDN-BLE framework in a BLE mesh testbed and provides an in-depth **experimental characterization** of its performance.
- It provides a **proof of concept** where a network congestion can be automatically detected and solved by the SDN-BLE framework.

This work extends the concepts introduced in [26] where a short overview of the SDN-BLE framework and an introduction to the proof of concept is given. This extension provides a deeper motivation of BLE as an all wireless SDN enabler and its main challenges and design needs; a complete overview of the whole framework with the inclusion of a centralized and distributed control plane; a new set of experiments in order to characterize the control plane operation in terms of energy consumption, reliability and overhead; and a performance characterization of the distributed and centralized control plane operation modes.

The remaining of the paper is structured as follows. Section 2 introduces the SoA of SDN for WSN and IoT, while Section 3 provides a brief introduction to BLE mesh and motivates its suitability for SDN-WSN. Section 4 describes the proposed SDN-BLE framework, and then Section 5 elaborates on its testbed implementation. Section 6 presents its experimental characterization and provides a proof of concept for this framework. Finally, Section 7 concludes this work.

2 RELATED WORK

Design and implementation of SDN enabled architectures for wireless networks has attracted much interest in recent years. Several surveys point out their aforementioned implementation challenges [8], specifically in the context of IoT [2, 17]. Some wireless SDN architectures for WiFi applied to Small Cells [11] and Cognitive Radio [38] have been published, but although they introduce improvements in spectrum context awareness by dynamically assigning SDN resources, there is no strict isolation of data and control information flows as they are mere logical wireless channels.

In the context of IoT and IIoT, great part of the literature focuses on IEEE 802.15.4 and Low-Rate Wireless Personal Area Networks (LR-WPAN), and most of them present a centralized controller entity that manages the network. The work in [1] introduces μ SDN as a lightweight, low-overhead architecture, while authors in [25] present Whisper as a management system to centrally control LR-WPANs. They are implemented in testbeds with commercially available IoT devices following the trends introduced by the IETF 6TiSCH (IPv6 over the TSCH mode of IEEE 802.15.4e) architecture. It defines centralized scheduling [35], which implements SDN concepts to provide spatial and frequency diversity within IEEE 802.15.4-2015 networks. Another popular approach for 802.15.4 is using the OpenFlow standard. For example, Sensor-Openflow [21] presents an SDN controlled MAC protocol which is able to robustly control a network with varying number of nodes. Some other works focus on synchronized flooding for 802.15.4, such as [9] where Glossy is used as a mesh protocol, but this flooding approach is still less popular than the TSCH version.

Regarding IP over low power networks, SDN for 6LoWPAN is a very popular research topic, but requires additional compression to not overload the resource constrained nodes. The works presented in [1, 23] show that the additional control overhead can be reduced while still having low end-to-end latency for data packets, but most of the solutions of SDN for 6LoWPAN focus on implementing SDN for RPL exclusively. Additionally, diversity in centralized versus distributed control plane approaches is missing on these solutions, and the isolation of the data and control flows is sometimes unclear. In fact, ensuring this separation while guaranteeing different QoS constraints is still not well understood in the context of wireless SDN [2, 17].

Moreover, when focusing specifically on SDN for BLE the only previous work published is BLESS [36], an architecture for creating wide area IoT service slices that use BLE devices at the edge. BLESS is a SDN controlled switch node that is transparently inserted between two communicating BLE devices and allows policy based switching between them. Therefore, it presents an introduction to a SDN-BLE enabled data plane where different policies can be transmitted over the air to reconfigure two nodes. No control plane implementation or discussion on how to ensure isolation of flows and architecture reliability and scalability is given, and reconfiguring BLE nodes comes at the cost of introducing additional ones. It also does not cover BLE mesh, and in fact to the best of the authors' knowledge there is no current proposal for SDN based on BLE mesh in the literature.

Finally, regarding the selection of Trickle and FruityMesh as mesh protocols the majority of the literature focuses on the former as it is part of the IETF's MPL protocol [15], which enables IPv6 multicast forwarding in Low-Power and Lossy Networks (LLNs). However, they again focus on 802.15.4 [34] networks. Regarding FruityMesh, a few recent works merely focus on characterizing its performance [16, 27], but to the best of the authors' knowledge this is the first work that proposes a hybrid BLE mesh scheme using both mesh protocols in parallel.

3 BLE AND MOTIVATION FOR SDN-BLE MESH

This Section presents an overview of BLE and BLE mesh in 3.1 and then motivates why this technology is a suitable candidate for an all wireless SDN-WSN framework in 3.2.

3.1 BLE and BLE Mesh Overview

BLE is a technology that operates in the 2.4 GHz ISM band, with the physical bandwidth split into 40 orthogonal channels of 2 MHz. There are two different ways of transmitting data: *advertising mode* and *connection mode*, with 37 channels reserved for data and 3 for advertising, although options for different bandwidths and advertising channels are available in the recent Bluetooth 5 specification [3]. Advertising channels are used for discovering services, broadcasting and initiating a connection between two nodes, while data channels are used for exchanging data once this connection is established.

BLE devices can then carry out the entire communication exchange solely in advertising mode by broadcasting packets in a round-robin scheme on the three advertising channels, or they can establish a connection and pseudorandomly hop on the data channels in a Master-Slave scheme. Advertising mode does not use any acknowledgement scheme to ensure reliable packet delivery, but the process is fast due to the lack of connection establishment. Connection mode, on the other hand, implements acknowledgement and retransmission of packets in order to guarantee message delivery and the two nodes have synchronized wake up timers. During the so called Connection Interval (*CI*) the nodes wake up, exchange a packet and go back to sleep, repeating the same process during the following *CI*s. This scheme increases the delay of the transmission but reduces the energy consumption of the nodes as they are sleeping most of the time.

The recent BLE mesh technology [4] is a flooding mesh protocol in which devices are continuously scanning and rebroadcasting packets. When a packet is received, its associated Time to Live (*TTL*) field is decremented by one and the packet is immediately rebroadcasted if its $TTL > 0$ and the packet has never been rebroadcasted by the node before. This flooding protocol ensures low end-to-end delay, but may cause excessive overhead [27] as it does not offer a way of controlling the number of rebroadcasts in the network. Additionally, having the nodes scanning all the time ensures that packets are never missed, but largely increases energy consumption. A partial solution is to actually allow the nodes to go into a sleeping state and introduce their associated Friend Node, which is a node that stays listening to the channel all the time and relays the messages lost while sleeping. However, this merely shifts the increased energy consumption versus reliability problem, rather than solving it. On top of that, the standard does not provide a clear specification on how to adequate this sleeping time to satisfy network requirements.

3.2 Motivation for SDN-BLE Mesh

The two main concerns for designing a wireless SDN-WSN framework are then respecting the isolation between the data and control information flows and ensuring the reliability of the control plane. BLE offers a unique opportunity to satisfy these two requirements since:

- (1) The **isolation** between data and control information flows can be satisfied by exploiting the inherent separation between advertising and connection modes that BLE provides. The definition of isolation used in this work means that the data and control information flows travel over different packets, paths, timeslots and frequency channels, and additionally the nodes have different dedicated state machines associated to each transmission mode and they are mutually exclusive. The network could be then effectively split into two isolated slices.
- (2) Both network slices offer different mechanisms to ensure **reliability**. Packets sent over connection mode use acknowledgement and retransmission mechanisms, while packets sent over advertising mode can be rebroadcasted by other nodes. Of course, both slices present a trade off between reliability, delay and energy consumption, but their parameters can be fine tuned independently to satisfy network requirements.

However, custom BLE mesh protocols present in the SoA use either flooding strategies, with packets broadcasted on the advertisement channels and using rebroadcasts as a propagation mechanism [4, 37], or use routed messages over the data channels, where a path can be constructed between a source and a destination but has to be maintained [22].

In [27] a comparison of these two mesh schemes for BLE is presented, where a flooding scheme is shown to reduce end-to-end delay while increasing the energy consumption of the nodes and a routing scheme is shown to behave in the opposite way. Therefore, each mesh scheme will be beneficial depending on the kind of traffic that is used. With these results in mind and taking into

Fig. 1. SDN scheme. The DP contains the nodes, while the CP manages the former based on requirements from the AP. The three layers exchange information through the Northbound and Southbound APIs.

account the requirements imposed by the SDN paradigm, it is clear that a custom implementation that includes both routing and flooding needs to be designed in order to allow SDN for BLE mesh.

Note however that while BLE mesh can be an adequate candidate for a SDN-WSN framework given the aforementioned reasons, the characteristics of the technology impose some other challenges for its implementation. The main one considered in this work is the fact that only two different network slices with different QoS requirements can be established in the network (the flooding and routing slice using the two transmission modes provided by BLE), but additionally ensuring concurrent streams of data with different requirements within each slice is not straightforward. Moreover, as the flooding slice is by definition a connectionless scheme, in general no network maintenance is needed and the mesh protocol chosen for this slice will not generate additional overhead (as in the case of Trickle for the proposed framework). However, the routing slice represents a connection oriented scheme and routing protocols usually need some network maintenance to perform properly (as in the case of FruityMesh for building the network tree topology). Therefore, in general three types of traffic will coexist in the network: data, control and maintenance, while BLE only allows for two wireless network slices. In this context, the proposed solution is to prioritize the data and control flows over the maintenance flow in order to ensure that the latter one does not interfere over the former two. Both data and control can still be isolated from each other, as they travel over two different network slices and the behavior of one slice does not affect the other one, as described in the next section.

4 PROPOSED SDN-BLE FRAMEWORK DESIGN

This section presents the proposed SDN-BLE framework. Section 4.1 gives a brief introduction of the SDN paradigm, while Sections 4.2 to 4.7 describe the design of all the framework layers.

4.1 SDN Overview

Generally, any SDN infrastructure is divided into three different layers, as depicted in Figure 1: a Data Plane (DP), a Control Plane (CP) and an Application Plane (AP). The DP contains all the nodes in the network, which merely create and forward packets. The CP allocates the intelligence of the network and acts as the intermediary between DP and AP: it is responsible to control the hardware in the DP and reconfigure the individual parameters of the nodes, so end-to-end network behavior can be optimized according to application requirements coming from the AP. This intelligence may be centralized into one main entity or distributed over a few ones. Finally, the AP provides all the necessary tools to monitor application performance and instantaneously delivers this information and requirements to the CP for fast network adaptability and robust policy making.

Several application programming interfaces (APIs) are in charge of intercommunicating these layers. The Southbound API carries information from the DP to the CP in the form of reports about the status of each node (such as buffer occupancy or routing tables) and conversely delivers

Fig. 2. Proposed SDN implementation for BLE mesh networks with a hybrid mesh DP where data packets are routed through connected nodes and control packets are broadcasted. Every BLE sink is managed by an associated gateway and two operation modes for the CP can be used: centralized on the server or distributed over the multiple gateways. In both cases the AP resides on the server.

control commands from the CP to the DP (such as routing decisions or channel selection). The Northbound API carries abstracted information of the network from the CP to the AP and, vice-versa, delivers end-to-end requirements from the AP to the CP. Additionally, East and Westbound APIs interconnect different modules of the CP that may be distributed over several entities.

Detailed information on the structure of the SDN paradigm can be found in [32]. In the sequel, the design of the aforementioned SDN elements in the proposed SDN-BLE infrastructure is given.

4.2 Hybrid mesh Data Plane

4.2.1 Routing and flooding mesh schemes in the Data Plane. The proposed SDN for BLE mesh framework is depicted in Figure 2. The DP is formed by the BLE nodes creating a hybrid BLE mesh network that uses both routing and flooding slices in order to respectively deliver the data and control plane information flows to the sinks, which are managed by associated gateways. The CP can be configured in two operation modes: distributed over the several gateways or centralized on a server. This flexibility in operation can be used to further increase its reliability and scalability. Finally, the AP resides in a central server.

In [27], a hybrid BLE mesh scheme with two isolated and independently reconfigurable mesh protocols is presented: a flooding scheme based on Trickle [19] and a routing scheme based on Fruitymesh [22]. While assignment of data and control plane information flows can be done on any of the two schemes, the proposal of this work is to have the intermittent control plane information flow assigned to the flooding slice and the application data flow to the routing slice.

A short characterization of each mesh protocol [27] is needed before justifying this proposal. A flooding slice over advertising channels is suited for extremely urgent data, multicast information and/or to serve mobile nodes, while a routing slice over data channels is suited for long lived multihop data streams. The reason is that a *flooding slice presents extremely low end-to-end delay*, and given the rebroadcasting mechanism used for packet dissemination it allows to exploit all possible paths from source to destination and ensures all nodes receive the same information. On the other hand, a *routing slice guarantees both low energy expenditure*, since the nodes have synchronized wake up and sleep patterns, and *reliable message delivery*, since it implements acknowledgement and retransmission of packets.

With this characterization in mind, the application data information flow represents a steady and periodic stream of data (where the node generates or forwards a packet every periodic event

Fig. 3. Node architecture and behavior over time.

denoted as T_{DP}), therefore requiring a reliable and energy efficient scheme in order to extend the life of the system and ensure packet delivery. In this regard, a routing mesh scheme such as Fruitymesh [22] is more useful for propagating application data through the mesh. In the proposed SDN-BLE framework this translates into a connected multihop DP mesh network with each node having multiple slaves and one master in a tree shape, with the sink on top of it. On the other hand, the control plane information flow represents a sporadic and bursty type of data that needs to be delivered to and from the control plane with guaranteed low delay in order to obtain real time network status information (every periodic event denoted as T_{CP}^{UL}) and over the air node reconfiguration (whenever such event is triggered in a reactive way, expressed as T_{CP}^{DL}), respectively. Therefore, a flooding mesh scheme such as Trickle [19] is suited for this type of traffic.

4.2.2 Trickle in the flooding network slice. The selection of Trickle as a flooding scheme for the control plane information flow in this SDN-BLE mesh implementation instead of the official BLE mesh protocol is justified by two main reasons. First, the CP links carry signalling information and thus have to keep the control overhead to a minimum, so the overall performance of the application data flow is not affected. Second, this control overhead needs to be exchanged regardless of the presence of application data being actually transmitted, so the additional energy expenditure for the nodes that it imposes needs to be minimal. With these reasons in mind, the official BLE mesh protocol represents a mere flooding scheme which does not offer mechanisms to effectively reduce energy consumption and network overhead. Trickle, on the other hand, inherently presents configurable mechanisms to suppress excessive rebroadcasts (by refraining the node from transmitting a packet already overheard more than a certain k times) in the network and control the periodicity of such transmissions (by transmitting at a random timestamp T_{ADV} within an interval that becomes exponentially longer from I_{MIN} to I_{MAX} when no new messages are overheard) [19].

4.2.3 Isolation of data and control information flows. An overview of the proposed architecture for the nodes is given in Figure 3a. The isolation of both information streams is justified by using these

two types of mesh schemes, having separate buffers for each flow and additionally using mutually exclusive timeslots and state machines, as the DP information flow packets are transmitted inside their respective *CI*s (where only one single connection to a master and up to S connections with slaves are allowed) and the CP information flow packets are transmitted at T_{ADV} . The relationship between T_{DP} and the different *CI* determines then the occupancy of the buffers allocated to the DP information flow, and, similarly, the relationship between $T_{CP}^{UL} + T_{CP}^{DL}$ and T_{ADV} the occupancy of the CP information flow buffer. As stated in the last section three types of network flows need to coexist in these two network slices. In the proposed architecture the data plane information flow clearly goes over the routing slice, while the control plane information flow clearly goes over the flooding slice. The network maintenance flow (used by the routing protocol) generally uses broadcasting as a mean of propagating information over the network, so it should be allocated in the flooding slice. However, in order to ensure that it does not interfere with control plane activity, a priority buffer is implemented for all advertising activity, so the maintenance is only performed over the flooding slice whenever there is no other control plane activity. This priority scheme is an adequate solution for the problem, as is able to fit the three network flows in the two network slices offered by BLE and only the maintenance flow is affected by the other two. In practice, different routing protocols will need different maintenance requirements, so the control plane activity will affect them in greater or lower extents, but the majority of them only need an initial exchange of messages in order to build the topology and then lower their maintenance to a low rate. Given that the control plane activity is also sporadic, in the proposed framework this is not harmful for FruityMesh maintenance, although this inherent limitation of BLE needs to be carefully addressed by the network operator in other deployments using other routing protocols.

Essentially, this hybrid BLE mesh architecture makes use of a time sharing scheme of the BLE forwarding resources, in a connection-based, flooding or sleep state, where all of them are mutually exclusive. The behaviour of the DP information flow is similar to regular BLE connected mode, but instead of sleeping between connection events the node remains scanning and broadcasting CP information flow packets. Therefore, all CP activity comes at the cost of higher energy consumption. To reduce its impact, a duty cycling (D) approach is introduced: $D = 0$ means no control messages are sent and the node goes to sleep between connection events (regular connected BLE behaviour); $D = 1$ means that the node remains scanning and broadcasting control messages for the whole interval between connection events and never goes to sleep (similar to the official BLE mesh protocol). The duty cycle process is implemented as a *coin toss*: any time the node is executing a Trickle based event, a random number R is generated from $U(0, 1)$, and if $R \geq 1 - D$ the event continues normally. If not, the event is suppressed and the node transitions into an idle state, potentially queueing the packet in case it was going to be transmitted. This scheme deviates from the classical duty cycling approach in which the node remains on and then off for a constant amount of time, which may result in higher packet losses due to the random nature of Trickle transmissions. As the nodes only have a single network interface, this mutually exclusive time slotting is needed along with the different frequency channels and packets used in order to ensure their isolation. Figure 3b shows the parallel flow of DP and CP packets with a duty cycle of $D = 0.5$ as an example. Note that the duty cycle value D has no impact over DP activity.

4.2.4 Reliability of the control plane information flow. Having covered the first SDN requirement of isolation of data and control plane information flows, the reliability of the CP needs to be addressed. Initially it may seem counterintuitive to rely on the flooding based slice for the control plane information flow, as it does not make use of packet acknowledgement or retransmissions. However, as the routing slice is a timeslotted contention based scheme, retransmissions incur in an increased delay and buffer occupancy, which may eventually result in congested links with

packets being dropped due to buffer overflows. The flooding scheme, by using rebroadcasts as a propagation mechanism, is not subject to any predefined path and therefore avoids congested links in the network. Additionally, if initially a node does not overhear a packet another node may rebroadcast it, increasing the chance of the first node to receive it this second time. This means that if enough nodes rebroadcast the packet, reliability can be ensured, without the cons of the increased delay and congested links that the routing scheme generates. By having complete freedom to configure the parameters of the flooding slice independently from the routing slice, the number of rebroadcasts and rebroadcasting nodes in the network can be easily tuned to make this control plane information flow as reliable as wanted in terms of packet delivery, of course at the cost of increased network overhead and node energy consumption. This is extremely useful in sparse networks or in parts of the network where different clusters of nodes are only interconnected by one or few physical links. In such scenarios, the duty-cycled flooding scheme could initially fail if the node on one end of this link would not receive a control packet (so this stream could not propagate to the remaining part of the network) and there were no more neighbors to rebroadcast such packet. In this situation, it is important that the flooding protocol used ensures that there are enough rebroadcasts in the network to allow proper message dissemination. In the case of Trickle, each packet is rebroadcasted for a certain k times in each single-hop range of M neighbors, and this load distribution is shared uniformly among the neighbors when the reliability is high (so each node will rebroadcast the packet k/M times) but otherwise the nodes that did receive the packet are in charge of rebroadcasting it more times so their neighbors receive it (with only one node rebroadcasting it k times in the worst case). Therefore, the value of k can be tuned differently for different parts of the network to ensure the reliability of the flooding scheme, unlike in the official BLE mesh implementation where each node rebroadcasts a packet only once. Finally, the diversity in terms of paths and transmission times that the flooding slice offers, combined with the frequency hopping scheme that BLE provides, ensures reliability against interference.

4.3 Southbound API

The Southbound API manages the control plane information flow between the DP and CP and can be divided into its uplink (from the DP to the CP) and downlink (from the CP to the DP) components. The uplink delivers periodic monitoring about the status of every node in order to obtain global information of the network, so it propagates from the nodes to the sinks. The downlink sends reconfiguration messages to adapt the behaviour of the nodes according to the policy created by the CP once it has this status information and the requirements of the AP, so packets travel from the sinks to the target node. In both cases the sinks deliver/receive these CP packets to/from its associated gateway, as explained in the next subsection.

4.3.1 Southbound API uplink. Figure 4a shows the metrics reported in the Southbound API uplink considering a BLE node that has one master and three slaves as an example. This status information summarizes the condition of the node in the time elapsed since its last transmitted data packet and is reported periodically every T_{CP}^{UL} , where generally $T_{CP}^{UL} > T_{DP}$. Of course, it can be decreased down to $T_{CP}^{UL} = T_{DP}$ and then a status information packet will be generated for every data packet generated by the node, although this can have a higher impact in terms of network overhead. The BLE node identifies itself by including its NodeID in each status message and specifies whether it is a sink, a static or a mobile node (where static nodes do not have battery limitations and are allowed to be in the upper layers of the tree, while mobile nodes can change their position in the network freely and therefore occupy the leafs of the tree). The node then appends the basic communication statistics between itself and its master, such as the master NodeID, the received average RSSI, the CI of this link and the average buffer occupancy. This exact information is also included for all the

Fig. 4. (a) Example of uplink Southbound API status report messages, containing BLE specific fields that enable the CP to infer the status of the DP. (b) Example of downlink Southbound API parameter update message from CP to DP, reconfiguring local node parameters to satisfy end-to-end performance requirements.

connected slaves, and additionally the node appends its battery level and transmission power and finally timestamps the status message.

4.3.2 Southbound API downlink. Southbound API messages in the downlink, on the other hand, contain reconfiguration parameters for the node, as exemplified in Figure 4b. This over the air parameter reconfiguration is used to individually tailor the performance of the nodes in order to adapt to changing network conditions and meet end-to-end AP requirements, and are delivered to the sink whenever the CP creates them (event triggered at T_{CP}^{DL}). The message is addressed to a specified node given by the NodeID and includes the parameter to modify and towards which connection should be modified (if applicable), the desired new value and a timestamp. It may seem overkill to flood the entire network to only reconfigure one node, but in practice when the network needs to be reconfigured due to suboptimal performance many nodes are responsible of this situation. The Southbound API downlink packets gather reconfiguration instructions for a set of nodes rather than sending an independent message for each node. As an example, consider a situation where the data plane is congested with a certain node acting as a bottleneck. The SDN-BLE framework will generate a reconfiguration message to that node so its CI with the next hop is decreased, but then the following nodes in the path towards the sink also need to decrease their CI so they do not become a new bottleneck in the network. This is a direct consequence of mesh networks, where nodes closer to the sink need more bandwidth to forward the cumulative traffic of all the previous part of the network along with their own generated traffic [14], and it is shown in the PoC of Section 6.3. The SDN-BLE framework will then generate a unique packet addressing all the necessary NodeIDs.

Finally, Table 1 summarizes the main parameters that can be reconfigured in the node, where both flooding and routing network slices can be tuned independently and changing their parameters will have an impact on their reliability versus energy consumption versus end-to-end delay.

4.4 Control Plane

The CP contains all the control logic of the BLE mesh network. Once the DP and CP information flows arrive at the BLE sinks they forward their received routing-based data packets and Southbound API uplink flooding-based packets to their associated gateways through a wired serial interface, which is also used for receiving the Southbound API downlink messages that later on propagate through the network to the target node. Two modes of CP operation are then possible: in the

Table 1. Over the air reconfigurable parameters of the nodes.

General Parameters		Trickle Parameters		FruityMesh Parameters	
P_{TX}	Transmission power	I_{MIN}	Min Trickle interval	S	Max slave connections
B	Max buffer size (per buffer)	I_{MAX}	Max Trickle interval	CI	Connection interval
D	Duty cycle	k	Rebroadcast threshold		
T_{DP}	Data packet generation period				
T_{CP}^{UL}	Status packet generation period				

centralized mode the control logic resides on a central server, while in the distributed mode it is spread over the different gateways controlling the sinks.

4.4.1 Centralized CP. When a fully centralized CP operation mode is selected (which is the approach followed classically in SDN [18]) the gateways may merely transmit all data and control packets that are received to the central server and all end-to-end information is stored and processed there. This operation is performed over a WiFi backbone which is not resource constrained and not part of the resources managed by the BLE mesh network. In this operation mode the server becomes the centralized CP and AP entity of the framework and this allows the Northbound API to be just a logical channel between the two, thus performing all this information exchange for free, which translates into a simpler and faster system. However, this entity is the single point of failure of the system and additionally needs to have enough resources to process and manage all nodes of the DP, which may arise scalability issues for dense networks. In this regard, both reliability and scalability of the CP can be compromised. This operation mode is shown in Figure 5a.

4.4.2 Distributed CP. When a distributed CP operation mode is selected the gateways are in charge of executing the control logic. In this approach, they obtain partial information about the status of the network, corresponding to the partial view that each associated sink has access to, and then share with each other their gathered data in order to guarantee that they eventually obtain global end-to-end information. This information exchange is done through the East and Westbound APIs. The server then becomes just the AP entity of the system, and the East and Westbound APIs, as well as the Northbound API, make use of the WiFi backbone isolated from the BLE mesh resources. This approach ensures some level of redundancy as there is no critical point of failure in the CP and provides a more scalable solution for dense networks, as the DP can be divided into different clusters that can be locally managed. This operation mode is depicted in Figure 5b.

To conclude, the flexibility of the centralized or distributed modes for the CP and the use of a different backbone is a key element to address the reliability and scalability of its implementation in the proposed SDN-BLE framework. Depending on the size of the network, topology and overall network conditions, one or the other mode of operation may be more suitable.

4.4.3 CP controllers. In any case, the controllers in the CP have two main tasks. The first one is receiving the status information of the nodes and, based on the requirements from the AP, designing individual node reconfiguration policies to meet such demands. The second one is providing the AP with an abstracted but functional model of the network. Note that the inclusion of the gateways acting as CP controllers instead of the BLE sinks is needed in the framework. This is justified by two reasons: as the BLE sinks only have one network interface using BLE as its technology, another device that interconnects the data plane with the CP controllers and the AP over a different backbone is needed so this traffic does not interfere with the BLE traffic; and additionally, as the BLE sinks are battery and computationally constrained devices they cannot perform the more processing intensive tasks of the CP controllers.

Fig. 5. (a) The gateways forward all DP and CP information flows over the WiFi backend, and both are delivered to the server through a wired connection. The application data packets are delivered to the AP, while the CP stream is delivered to the CP controllers. (b) The CP controllers are replicated on the gateways, which exchange the partial view of the network that their associated sinks deliver them in order to obtain global end-to-end information.

The specific design of the CP controllers is tightly related to the actual implementation of the SDN architecture. For this work, four different controllers are proposed: the *Dynamic Node Controller*, the *Meta Data Parser*, the *Network Topology Builder* and the *Anomaly Detector*.

As an overview, the Network Topology Builder determines the shape of the network based on the Southbound API uplink packets of every node, draws the topology and determines the NodeID, device type and status of every node (as in correctly functioning or malfunctioning). It also sends this information to the AP via the Northbound API uplink so the application can see an overview of the network. The Meta Data Parser extracts information directly from the data packets and feeds it to the CP using the Northbound API downlink, so the CP has direct information on the current performance of the network and additionally the application performance requirements. With this information the Anomaly Detector and Dynamic Node Controller can tailor individual node performance. The Anomaly Detector makes use of the information collected by the Meta Data Parser to determine whether the expected amount of data is received from each node and, if not, whether this is due to sensing errors, node failure or link congestion based on the history of Southbound API uplink status packets. Finally, the Dynamic Node Controller is responsible for manipulating each node through the Southbound API downlink in order to keep reliability high and avoid congestions, based on the alarms triggered by the Anomaly Detector. Detailed implementation details of these controllers are given in Section 5.3.

4.5 East and Westbound APIs

When the CP is operating in distributed mode the East and Westbound APIs are responsible for the CP information flow exchange among gateways. The gateways collect the Southbound API uplink packets that their associated sinks deliver to them and encapsulate this information into a multicast WiFi stream. This way, the rest of gateways are also able to obtain the status report stream from the other BLE nodes and complete their global vision of the network. Additionally, the multicast operation allows to use a single transmission in order to exchange information to all gateways, therefore decreasing overhead in the backbone. Since the different CP controllers are replicated in each one of the gateways, once they obtain end-to-end information they are able to

Fig. 6. (a) East and Westbound APIs used to encapsulate and deliver a BLE data stream between two different BLE mesh clusters over the WiFi backbone. The process is transparent to both BLE transmitter and receiver. (b) Northbound API with abstracted network view in the uplink and real-time performance and application requirements in the downlink.

locally detect network anomalies and reconfigure the nodes responsible for such anomalies in the same way that the server does when a centralized CP operation mode is used.

Additionally, the East and Westbound APIs can be used to effectively split the whole BLE mesh network into several BLE mesh clusters that are interconnected with each other over the long range WiFi network created by the gateways. In this sense the communication among BLE nodes from different clusters can be done transparently irrespective of distance, as the gateways can forward the data plane packets until they reach the target sink, as seen in Figure 6a. This solution can considerably increase the range of the BLE mesh without the need of adding intermediate BLE nodes, which is the only alternative present in the official standard and is a solution that may require too many nodes for the range that it can actually cover.

4.6 Northbound API

The Northbound API acts as the communication layer between the CP and AP, and like the Southbound API it can be divided into its uplink and downlink components. The Northbound API uplink provides the AP with an abstracted view of the network so the application can have an overview of the status of the system, while the Southbound API delivers application requirements and performance metrics so the CP can perform effective network policing.

4.6.1 Northbound API uplink. The Network Topology Builder reports the topology of the network, device type and status of every node through the Northbound API. This information is sent every T_{NB}^{UL} . When a centralized CP operation mode is selected, as both the AP and CP reside in the central server this information does not involve any additional network overhead. When a distributed CP operation mode is used, each gateway transmits this information every T_{NB}^{UL} , but their clocks are not synchronized to avoid congestions.

4.6.2 Northbound API downlink. The Northbound API downlink provides the CP with performance statistics extracted from the application data packets received by the BLE sinks and forwarded to the AP. For every received packet, a meta-data parser generates a summary of the current performance and application requirements as seen in Figure 6b. It is important to note that this characterization of the behaviour of the network is performed based only on the application data packets and no explicit control information is necessary, so again when the CP is centralized this is done for free at the central server. Having the performance summary and the application requirements, together with the status of the nodes and topology of the network, the CP is able to run the Anomaly

Detector and create individual node reconfiguration policies. This closes the loop and allows for real-time monitoring of end-to-end performance, which is absent in current low energy WSNs.

4.7 Application Plane

Finally, the AP residing on the server contains several agents: a specific *Application Data Parser*, used to extract the performance of the DP that is fed back through the Northbound API downlink to the Meta Data Parser; a *real time GUI (Graphical User Interface)* used to monitor the performance of the whole network and visualize the content of the received application data packets; a *cloud connector software* that pushes the real time data stream and network status information into any remote server in order to make this information available through the internet; and a *database logger* to save all the packets transmitted over the network by all nodes.

5 SDN-BLE MESH TESTBED IMPLEMENTATION

This section presents detailed explanations on the implementation of the proposed SDN-BLE mesh framework. Section 5.1 introduces the automated BLE hybrid mesh testbed used for the implementation. Section 5.2 describes the implemented DP and the two mesh protocols used, while Section 5.3 provides detailed explanation on the controllers of the CP.

5.1 Testbed

The proposed SDN-BLE framework has been implemented in a low-cost automated BLE hybrid mesh testbed. It consists of a set of BLE nodes, single board computers, a network switch, a PC server and several gateways, as shown in Figure 7a. The BLE nodes, gateways and server conform the SDN-BLE framework layer as described in the last Section, while the single board computers provide an additional testbed management layer. Reference [28] summarizes the hardware used and the cost of the testbed.

The testbed is deployed over an entire floor of a regular office environment, formed by a main control room, seven office rooms and a corridor. The control room is 6x9m contains 12 BLE nodes, one 802.11 board (with its associated BLE sink) and the PC server. The rest of the rooms allocate 28 BLE nodes and one 802.11 board, plus its associated BLE sink. All devices are hanging from rails 2.5m above the floor creating a grid where the average separation between two nodes is about 2m. The nodes are placed facing down, although their antennas are omnidirectional. The 802.11 boards are equipped with 2.4 and 5 GHz omnidirectional antennas, and are placed in vertical polarization.

The selected single board computer is the Hardkernel Odroid C2 [12], which provides an Ethernet interface and four USB ports. Each Odroid has four BLE nodes connected in order to power them (although regular coin cell batteries can also be used) and to establish a terminal session through the USB port so the transmitted and received packets are logged into the database on the server through the wired backbone. This is done for offline network forensics purposes in order to extract the delay and reliability of each link in the network, but note that this has absolutely no effect on the performance of the SDN-BLE framework and does not need to be implemented in a real life deployment where mobile wireless nodes are used. The gateways also log all received packets, both BLE and WiFi, so accurate statistics of the data and control plane traffic flows can be obtained.

The selected BLE nodes are the Nordic Semiconductor nRF52 development boards [30]. They are flashed with a full BLE stack (known as the Softdevice, which is a closed implementation provided by Nordic Semiconductor) and the proposed hybrid SDN architecture with the routing and flooding mesh protocols of Figure 3a is developed as an application on top of it. Compared to the work introduced in [28], all control and data packets are wirelessly transmitted to the sink, following the SDN-BLE framework previously described.

Fig. 7. (a) Architecture of the testbed with the BLE-SDN framework layer and the testbed management layer. The single board computers provide a terminal session with the nodes for powering and logging of transmitted and received packets. These are written in a database for network forensics purposes. (b) Picture of the testbed with the central server with its gateway and the nodes hanging from ceiling rails.

The 802.11 gateways are MikroTik RB912UAG-2HPnD Routerboards [24], which provide both a wireless interface for networking and a USB interface where the sinks connect to so the received packets are printed over a serial connection. They provide connectivity to the AP over a WiFi backbone.

5.2 Data Plane

The architecture for the BLE nodes of Figure 3a is implemented as an application in C++ and flashed on top of the BLE stack, using the Softdevice provided by Nordic Semiconductor. As a result of the Softdevice used, in the current implementation a node can have only one master device but multiple slaves. A separate buffer is used for every connection, although due to the physical memory available on the nRF52 nodes, the more connections are used the shorter the queue of each one needs to be. A typical value set in the framework is $S = 3$ slave connections and a buffer size of $B = 512$ Bytes for each connection, which can store 25 mesh packets.

5.2.1 Trickle based flooding slice. As introduced in [27], two kinds of parameters are used in Trickle: fixed and variable. The fixed parameters include the redundancy constant k and the limits of the Trickle Interval I_{MIN} and I_{MAX} . The variable parameters include the actual Trickle Interval I , the redundancy counter c and the transmission time event t .

The operation of Trickle is as follows. The Trickle Interval I is divided into a listen-only period from $[0, I/2)$ followed by a transmission period from $[I/2, I]$, in which t is randomly generated from $t \in U(I/2, I)$. Whenever the latest message in the network is overheard c is incremented by 1, and at time t the message is rebroadcasted if $c \leq k$ and otherwise suppressed. After the interval I expires, c is reset to 0 and its length is doubled (until it reaches I_{MAX}). Whenever the node overhears a new message the Trickle Interval is reset to I_{MIN} , repeating the whole procedure.

The main parameters that affect the performance of the control plane information flow are then the Trickle intervals I_{MIN} and I_{MAX} , which relate to how often packets are rebroadcasted, and the redundancy constant k , which relates to how many times a packet is rebroadcasted in the network. Figure 8a depicts the implemented Trickle based state machine, where the operation

Fig. 8. Simplified state machines for the implemented Trickle and FruityMesh protocols in the DP.

may be suppressed by the proposed duty cycle energy saving scheme (a feature not present in the original algorithm).

5.2.2 FruityMesh based routing slice. FruityMesh [22] is an open source implementation of a routing mesh scheme for BLE, based on neighbour only routing. A heavily modified version of FruityMesh is implemented in the SDN-BLE framework, and an overview of its different states is presented below, expanded from [5].

Initialization. When a node is activated it generates automatically a NodeID and a ClusterID, which are persistent after rebooting. The NodeID identifies the node in the network and the ClusterID identifies to which cluster the node is part of. At starting time the *Cluster size* is set to 1. As new nodes appear, the algorithm will force them to connect to bigger clusters until all nodes in the network are part of the same cluster and thus reachable by every other device.

Discovery Phase. After initialization, a node broadcasts its NodeID, ClusterID and its Cluster size as advertisements in the so called JoinMe packets (FruityMesh maintenance packets), and it scans for similar messages from its neighbours. This is continued until a route is set up. The discovery phase can have high to low broadcasting interval duration depending on the stability of the network: the longer the network remains stable, the longer the time between advertisements. This means that the maintenance of FruityMesh rarely interferes with other CP activity once the network is formed.

Route setup. After a predefined interval, each node determines to which cluster to connect to based on the received maintenance packets. The neighbours are ranked based on the Southbound API uplink status messages, and depending on the routing strategy (prioritize low buffers, low hop count towards a sink, etc) the node with the highest ranking becomes cluster head, so a new node may perform a handshake with it [29]. The connection of two nodes belonging to the same cluster is avoided as information on ClusterID is sent during the handshake, and as a result the whole network is loop free. If the ClusterIDs are indeed different, the node with the smallest cluster size connects to the bigger cluster.

Self-healing and Route Maintenance. As the network reaches stability and all the nodes are part of the same cluster, the discovery mode is switched to low, increasing the time between advertisements. If a connection is lost then the connection partner or the neighbours of the failing node switch to

Fig. 9. An example of network graph showing the BLE nodes represented by circles containing their NodeIDs. Sinks are represented in a darker color (NodeIDs 2442 and 3627).

higher discovery mode. As a self-healing mechanism, alternative routes are generated every time a node disconnects.

Routing. Routing is implemented in a neighbour-only fashion. No routing tables are present and each node is only aware of the nodes around it. When a sink is present, this information is broadcasted over the network. Each node then, when it has data to send, selects the neighbour with the smallest hop count to the sink and forwards the packets. Note that multiple sinks are allowed, even when they are managed by different gateways.

Figure 8b depicts a simplified state machine of the FruityMesh implementation. Data packets are exchanged in the *Connection* state, while scanning and broadcasting of JoinMe and other maintenance packets is done in the *Discovery* state.

5.3 Control Plane

Implementation details of the three main agents of the CP (the Network Topology Builder, the Anomaly Detector and the Dynamic Node Controller) are given next, as the Meta Data Parser is a mere logical channel from AP to CP with data packet information and application requirements. These are pieces of software programmed in Python and run either on the server or on the gateways.

5.3.1 Network Topology Builder. The Network Topology Builder builds a graph of the connections in the BLE network using the Southbound API uplink status messages given in Figure 4a, showing their NodeIDs, device types and connected nodes. Since in the current implementation each BLE node can have only one master, it is then trivial to build a directed graph with the control information obtained. Additionally, based on the information provided by the Anomaly Detector, the status of each node can be obtained and displayed in the Topology builder, where the user can visualize if there is an issue detected on a node (node failure or dead battery) or on a link (congestion between two nodes). Figure 9 shows an example of graph showing the nodes represented by circles containing their NodeIDs, where sink nodes are shown in darker color.

5.3.2 Anomaly Detector. The Anomaly Detector is built based on the probabilistic passive diagnosis model proposed by Liu et al. in [20]. This work uses a Bayesian Network to infer whether a node has failed to deliver data and differentiates between two possible failures: node and link failure. The former means that there is no data from a particular node (so it is down due to failure or dead battery), while the latter means that data is unable to hop through a specific link (due to channel impairments or buffer overflow). Note that the Anomaly Detector is only able to raise a

flag when one of these two anomalies is detected, but the actual failure can be obtained based on the history of the Southbound API uplink status messages: the evolution of the battery level versus a sudden failure for node anomalies and the evolution of buffer occupancy versus channel qualities for link anomalies. The Anomaly Detector requires the path between each node and its sink and the amount of data generated by each node in a certain window of time W . The former is obtained directly from the Network Topology Builder, while the latter is obtained from the Northbound API downlink. The Anomaly Detector produces reports of the anomaly probability $\mathbb{P}_A$ of each node comparing the difference of data throughput received in this window $R_{DP}(W)$ with its expected value $\mu_{R_{DP}}$ to a certain threshold δ :

$$\mathbb{P}_A(W, \delta) = \begin{cases} 1, & \text{for } \mu_{R_{DP}} - R_{DP}(W) \geq \delta \\ \frac{\mu_{R_{DP}} - R_{DP}(W)}{\delta}, & \text{for } \delta < \mu_{R_{DP}} - R_{DP}(W) < 0, \\ 0, & \text{for } \mu_{R_{DP}} - R_{DP}(W) \leq 0 \end{cases} \quad (1)$$

$\mu_{R_{DP}}(W = w) = \mathbb{E}\{R_{DP}(W = w - 1)\}$ (so the system has a memory of one window of time, although it can be increased to multiple windows) and δ is usually set to one standard deviation $\delta = \sigma_{R_{DP}}$. Note that reactive or proactive approaches can be obtained by fine tuning the observation window W , the threshold δ and the rate of Southbound API status reporting messages. Additionally, the current implementation only focuses on received packets, but including packet delays could increase the robustness the proactive approach.

5.3.3 Dynamic Node Controller. Finally, the Dynamic Node Controller receives the reports from the Anomaly Detector and sends, through the Southbound API downlink, configuration settings for the BLE nodes. If an anomaly report contains a probability of a sensing anomaly above a predetermined threshold $\mathbb{P}_A \geq T$ (usually set to $T = 0.7$) the node is considered faulty and a network operator might be notified. On the other hand, if the anomaly report shows a link anomaly, this can be due to a lack of connection entirely (the channel impairments are too high) or to a congestion (the node is a bottleneck). In the former case, obtained from the average RSSI values of the link, the transmission power of the node is increased. In the latter case, obtained from the buffer occupancies, the CI is decreased.

6 SDN-BLE MESH EXPERIMENTAL CHARACTERIZATION

This section provides an in depth experimental characterization of the implemented SDN-BLE mesh infrastructure. Section 6.1 presents a characterization of the CP in terms of duty cycle versus reliability and energy consumption, while Section 6.2 presents a characterization of the distributed versus centralized CP operation modes. Finally, Section 6.3 describes a proofs of concept where a network congestion is reactively detected and automatically solved.

6.1 Control plane characterization: reliability vs energy consumption vs overhead

6.1.1 Setup. In order to characterize the proposed Trickle based BLE mesh flooding slice and the effect of the duty cycle D , eight nodes are placed in the main control room and a connected mesh network is built. The network size for this experiment may seem small at first, but the increase in performance of adding more nodes than the ones considered is negligible as the results show a saturation behavior. All nodes are in range of each other, with a connection interval of $CI = 3$ s, a transmission power of $P_{TX} = 0$ dBm and a buffer size of $B = 25$ packets. Once the network is established, a transmitter and receiver pair is selected at random and 256 control packets are generated by the transmitter at a rate of $R_{CP} = 1$ packet/s. The Trickle threshold is set to $k = 8$ for all nodes, which means that each node will refrain from rebroadcasting a packet overheard already 8 times. The main reason for this choice is allowing the chance for each node to rebroadcast, since

Fig. 10. (a) Control messages Packet Delivery Ratio (PDR_{CP}) versus duty cycle (D) with varying intermediate rebroadcasting nodes, along with linear and sigmoid fits for single hop PDR_{CP}^{single} . (b) Number of rebroadcasts for each received control packet in the network (bars) and increase in control message Packet Delivery Ratio (ΔPDR_{CP}) (lines) versus network size for different values of the duty cycle (D).

the maximum number of rebroadcasts within a Trickle network assuming no loss of packets is in fact k [19]. This allows for comparison with the official BLE mesh protocol, as in this technology each node also rebroadcasts each received packet once. Statistics of the PDR for the 256 control packets are computed at the receiver with a duty cycle for all nodes (except the transmitter) ranging from $D = 0.1$ to $D = 1$, where 10 different realizations are set for every configuration to average out the results. The single hop packets received directly from the transmitter are taken into account, as well as the packets being rebroadcasted by the rest of the nodes.

6.1.2 CP Packet Delivery Ratio. Figure 10a depicts the results of PDR_{CP} vs D obtained, where lower duty cycle values translate into packets not being scanned and therefore lost. In order to analyze the effect of the duty cycle on the node the basic operation of the nodes from Figure 3b is taken into account, where the number of control packets transmitted per CI is directly modified by D . Since the CI chosen is rather high, given the relationship between R_{CP} and CI approximately 3 control packets are received per CI when $D = 1$, i.e., the node does not go into sleep mode. As a rule of thumb, neglecting the time dedicated to FruityMesh data packet exchanges, the expected number of unique control packets transmitted and received (assuming no loss due to channel impairments) in a single hop link within one CI window can be approximated by

$$N_{CP} \approx CI \cdot R_{CP} \cdot D \quad [\text{packets}], \quad (2)$$

which becomes more accurate the higher the value of CI becomes. Then, assuming independence of transmitting and receiving events and accounting for the loss of packets due to duty cycling the PDR can be approximated by simply

$$PDR_{CP}^{single}(D) \approx D, \quad (3)$$

which can be used as a first order approximation given its simplicity. When measured, the PDR shows a sigmoid behaviour, justified by the unsynchronized duty cycle timers and the normally

distributed random transmission times of the control packets (caused by the Trickle algorithm):

$$PDR_{CP}^{single}(D) = \frac{1}{1 + e^{-9.399(D+0.3307)}}, \quad (4)$$

where the coefficients are fit from the measurements.

Since the nodes have unsynchronized state machines, packets lost in this single hop link due to the receiver being in sleep state while they were being transmitted can potentially be picked up by other nodes, rebroadcasted at a later stage and eventually be received. Therefore, these expressions can be used as the lowest bound for a required number of control packets and CP reliability in the network, which can be increased by either increasing the duty cycle or the number of nodes. When the network size is increased, the network overhead increases due to the higher number of nodes rebroadcasting packets, and when the duty cycle is increased the energy consumption of the nodes is increased, due to the longer scanning times. A characterization of both strategies is given below.

6.1.3 Network overhead. Increasing node diversity has a negative impact in CP overhead (the number of times a packet is rebroadcasted in the network) and a decaying benefit in terms of PDR, as the chances of all existing nodes missing a packet and only being picked by a new one decrease with network size. This can be seen in Figure 10b, where the increase of PDR_{CP} when compared to the single hop case (ΔPDR_{CP}) is also presented. Adding more nodes with low or high duty cycle marginally increases the PDR_{CP} : the former because the new nodes also have a low probability of scanning the packet and the latter because packets are already received. Note that this increase in ΔPDR_{CP} is larger (and also more beneficial when taking into account the overhead introduced in the network as a side effect) for low values of D . However, the range of interest lies within intermediate values of D (corresponding to the center of the curve of Figure 10a), where duty cycle and node diversity can be traded and the increase of CP overhead can be controlled. Recall that the number of rebroadcasts in the network can also be controlled by modifying the Trickle threshold k , but its effect on PDR_{CP} and fine tuning strategies for different nodes in order to adequate the reliability of the control plane to global system requirements is left out as future work.

Therefore, the best compromise between PDR_{CP} and CP overhead is to use intermediate values of D and increase the network size. This also translates into a lower energy consumption for the nodes, as introduced below.

6.1.4 Energy consumption. The contributions of both D and CI are considered for computing the energy consumption of the nodes. Figure 11a shows the estimated lifetime L of a 1000 mAh capacity coin cell battery used to power up a node as a function of the duty cycle D and for different control packet transmission rates of $R_{CP} = \{0.1, 1, 10\}$ packets/s. Apart from the control packet activity, a bidirectional communication with one master and one slave is taken into account, generating data packets at a constant rate of $R_{DP} = 1$ packets/s with $CI = 1$ s, $P_{TX} = 0$ dBm, $V_{DC} = 3.3$ V and all packet sizes of 20 bytes.

This lifetime estimation can be obtained based on the nRF52 Online Power Profiler [31], which is a tool extracted from actual measurements conducted on the nRF52. It provides a model for the charge and current consumptions and length of several BLE events (connecting to a node, advertising packets, etc) and their sub-events (node startup, transmission, radio switching, scanning, packet processing, etc). Generally, the average charge consumption of an event $\bar{C}_{event}$ can be obtained by

$$\bar{C}_{event} = \bar{A}_{event} \cdot T_{event} \quad [C], \quad (5)$$

where $\bar{A}_{event}$ is its average current consumption and T_{event} is its duration, which is the sum of all the different sub-events. Considering the consumption per CI , four BLE events are should be taken into account: data exchange with a master and a slave nodes, broadcasting control packets and

Fig. 11. (a) Estimated node battery lifetime (L) versus duty cycle (D) for different control packet transmission rates (R_{CP}), along with inverse fit for $R_{CP} = 1$ packets/s, with $CI = 1$ s and 1000 mAh capacity. (b) Normalized time of each event (data, control, scan and sleep) relative to the Connection Interval (T_{event}/CI) for different values of its duration (CI) and control packet transmission rates (R_{CP}), with $D = 0.5$.

scanning and sleeping for the remaining time depending on the duty cycle value, as in Figure 3b. The estimated average current consumption per CI is then:

$$\bar{A}_{CI} = \frac{\bar{C}_{data}}{CI} + \frac{\bar{C}_{control}}{CI} + \frac{\bar{C}_{scan}}{CI} + \bar{A}_{sleep} \cdot \frac{T_{sleep}}{CI} \quad [A]. \quad (6)$$

Based on the nRF52 Power Profiler, the corresponding values are:

$$\begin{aligned} \bar{C}_{data} &= N_{DP} \cdot \bar{C}_{conn} = 14.98 \mu C & \text{with } T_{data} &= N_{DP} \cdot T_{conn} = 5.42 \text{ ms} \\ \bar{C}_{control} &= N_{CP} \cdot \bar{C}_{adv} = N_{CP} \cdot 11.65 \mu C & \text{with } T_{control} &= N_{CP} \cdot T_{adv} = N_{CP} \cdot 4.09 \text{ ms} \\ \bar{C}_{scan} &= N_{RX} \cdot \bar{C}_{RX} = N_{RX} \cdot 5.88 \mu C & \text{with } T_{scan} &= N_{RX} \cdot T_{RX} = (CI - T_{data} - T_{control}) \cdot D \\ & & \text{and } T_{RX} &= 2.96 \text{ ms} \\ \bar{A}_{sleep} &= 2 \mu A & \text{with } T_{sleep} &= (CI - T_{data} - T_{control}) \cdot (1 - D), \end{aligned} \quad (7)$$

where $N_{DP} = 2$ is the number of data packets exchanged in the CI ; N_{CP} is the number of control packets broadcasted in the CI and is given by (2); N_{RX} is the number of scanning events performed by the node and T_{RX} their duration; $\bar{C}_{conn}$ and $\bar{C}_{adv}$ are the charge consumptions for all the sub-events involved in a connection or advertising mode packet exchange; and T_{conn} and T_{adv} their durations, respectively.

The results of Figure 11a show a decreasing trend in battery lifetime L for increasing duty cycle D , where the contribution of the control packet rate R_{CP} can be neglected. An inverse fit can be applied to the battery lifetime when $R_{CP} = 1$ packets/s (which is a typical value in the testbed):

$$L(D) = 21.24 \cdot D^{-0.963} \quad [\text{days}]. \quad (8)$$

This is due to the fact that a high duty cycle translates into a large T_{scan} , which becomes a dominating term compared to T_{data} and $T_{control}$, especially when the value of CI is large. This effect is depicted in Figure 11b, where the normalized duration of each event relative to the duration of the CI (T_{event}/CI) for different values of CI and R_{CP} are given for a duty cycle value of $D = 0.5$. In fact, the contribution of $T_{control}$ is always the lowest one, even for high values of R_{CP} , which means a high control rate can be set in the network and a lower duty cycle can be achieved without

compromising the energy consumption and the reliability of the control plane, since increasing the number of nodes could increase this reliability and even if would be moderate the rapid status reporting packet generation of the nodes could still ensure that the longevity of the received control packets at the gateway would be low.

6.1.5 Comparison with BLE mesh. The current official version of BLE Mesh, present in the Bluetooth standard, relies on mere flooding to propagate packets over the network. As such, nodes need to be always scanning, which is equivalent to a duty cycle of $D = 1$. This translates virtually into a $PDR_{CP} = 1$ from Figure 10a (without taking into account packet loss from channel impairments), but also into higher network overhead from Figure 10b, as each node rebroadcasts each packet once is received. However, BLE mesh has only one mode of transmitting packets and isolation of data and control information flows is not possible, so these two types of traffic will interfere each other. Additionally, there is no randomization in the transmission timers of the nodes, which may lead to increased collisions. Regarding battery lifetime, Figure 11a shows an estimate of the performance of this technology, where current consumption is very high due to continuous scanning. In fact, the role of the Friend Node in BLE 5 [3] is necessary to avoid exhaustion of batteries. This is a trivial solution that does not allow fine tuning depending on application needs.

With the proposed SDN-BLE framework energy consumption and packet delivery ratio for the control plane information flow can be traded and fine tuned without a large increase in network overhead (as seen in the obtained results of Figure 10 and justified in Sections 6.1.2 to 6.1.4), which can additionally be controlled by modifying the value of the Trickle threshold k . As a result, nodes that are located in the core network could have high values of D , while edge nodes could decrease the value of D and the energy consumption of the network could be reduced, while ensuring reliability by increasing the network size.

For the hospital application considered, this translates into longer battery lifetime for the patients's sensors while still ensuring reliable message delivery, a critical achievement in such scenario. Concrete characterization on the scalability of the network for larger network sizes is left as future work, although a saturation behavior is already seen during these experiments.

6.2 Control plane characterization: distributed versus centralized operation modes

6.2.1 Setup. In order to characterize the distributed (DCP) and centralized (CCP) CP operation modes, a network of 18 BLE nodes in a LoS regular office environment where each node is in range of each other is set up. In the CCP operation mode all 18 nodes form a single network cluster and send their packets towards a common sink. The gateway associated to that sink receives all data and control packets and pushes this traffic through the WiFi backbone towards the server where the CP and AP reside. In this case, this gateway is then able to obtain the whole end-to-end network information. On the other hand, when the DCP mode is used, the network is split into two different clusters of 10 and 8 nodes respectively, where each cluster is isolated (so DP and CP packets from the neighbor cluster are discarded) and the nodes forward their traffic towards their respective cluster sink. In this operation mode, each associated gateway receives the partial data and control plane information flows associated with each cluster, so they need to exchange their respective control information in order to reach the same end-to-end knowledge. In both scenarios the default $CI = 100$ ms, with a data packet generated every $T_{DP} = 1$ s, the buffer size $B = 25$ packets, the transmission power $P_{TX} = 0$ dBm, the duty cycle $D = 0.5$ and the control packet transmission rate is swept $R_{CP} = \{0.1, 1, 10\}$ packets/s, so the time between control packets is $T_{CP} = \{10, 1, 0.1\}$ s, respectively. The measurement tests are repeated 5 times in order to average out the results. Statistics of the convergence time until the CP obtains full end-to-end network knowledge and the associated network overhead are given below.

Fig. 12. (a) Elapsed time until the CP converges to full end-to-end network knowledge for distributed (DCP) and centralized (CCP) operation modes using different control packet transmission rates (R_{CP}), along with statistics of the raw difference between them (Δt_c) and normalized by the time between control packets ($\Delta \bar{t}_c$). (b) Number of unique transmitted CP packets in the BLE mesh and the WiFi backbone by Gateway 1 and Gateway 2 (bars), along with the reliability (PDR_{CP}) of these two networks (markers), in the distributed (DCP) and centralized (CCP) CP operation modes for different control packet transmission rates (R_{CP}).

6.2.2 End-to-end network knowledge CP convergence time. Figure 12a depicts the number of entries received by the Network Topology Builder residing in the CP against the elapsed time for the three control packet transmission rates when the CP is distributed or centralized. When the number of unique entries reaches 18 this means that the CP has full end-to-end knowledge of the network.

When comparing the different control packet transmission rates, it is clear that faster rates translate into faster convergence times for both cases. The difference in convergence times between the DCP and CCP operation modes is defined as

$$\Delta t_c = t_c^{DCP} - t_c^{CCP} \quad [s]. \quad (9)$$

The normalized difference in convergence times is simply defined by dividing it by the corresponding time between control packets

$$\Delta \bar{t}_c = \frac{\Delta t_c}{T_{CP}}, \quad (10)$$

where its average value $\mu_{\Delta \bar{t}_c}$ and standard deviation $\sigma_{\Delta \bar{t}_c}$ are given in the figure.

When comparing the DCP with the CCP operation mode $\mu_{\Delta \bar{t}_c} < 1$ for all cases, which means that on average the difference in convergence times of the two lies within the time between two consecutive control packets. As a direct conclusion, both operation modes perform equally in terms of convergence time since the delay introduced by the high bandwidth WiFi backbone and the gateway processing is negligible compared to the delay introduced by the resource constrained BLE network. Note that this is also supported by the progressive decrease in $\mu_{\Delta \bar{t}_c}$ and $\sigma_{\Delta \bar{t}_c}$ for lower control packet rates.

In both cases it is clear that the main parameter that can influence the rate of convergence to full information is the control packet generation rate, at the cost of increased CP overhead. Therefore, given that the backbone of the network has enough bandwidth, a distributed approach may always be more suitable because a comparable performance can be obtained without focusing the whole

intelligence of the network on a single entity, which is beneficial for reliability reasons. Additionally, the scalability of both approaches for larger number of nodes may conclude that the distributed approach is more suited, although further research on this topic is left as future work.

6.2.3 CP reliability and overhead in the BLE mesh and WiFi backbone. Figure 12b depicts the number of unique control packets transmitted by the nodes in the BLE mesh and by the gateways in the WiFi backbone in the CCP and DCP operation modes. The definition of *unique* implies that when a node generates a packet all rebroadcasts of other nodes due to the flooding protocol are not taken into account, but retransmissions of this node due to this packet not arriving to the sink are considered. When CCP is used Gateway 1 forwards all CP packets to the server in the WiFi backbone, and when DCP is used Gateways 1 and 2 do so.

In DCP mode the overhead in the BLE mesh increases while it stays the same in the WiFi backbone, although it is spread over different gateways. This is because in this DCP scenario the size of the BLE network is smaller, which translates into a lower PDR_{CP} and thus more retransmissions are needed to effectively deliver all control packets to the associated gateways. Since the WiFi backbone has high reliability and packets are not lost, in addition to the multicast transmission mode used to spread information through the West and Eastbound API, the number of packets transmitted in the distributed and centralized case is exactly 18.

Moreover, the measurements show that the reliability of the CP PDR_{CP} is not affected by the control packet transmission rate R_{CP} (similarly to what is seen in Figure 11a with the battery lifetime), and based on all the results obtained indeed the only parameters that affect it are the duty cycle D and the network size. This can be seen on Figure 12b because only *unique* packets are taken into account, but higher control packet rates will of course translate into higher BLE overhead, which can, in turn, be lost completely if the duty cycle is not high enough.

As a conclusion, a DCP operation mode is beneficial for the BLE mesh when the network size is large, which is indeed the case in which network management can become challenging and distributing the CP over several clusters is appealing. This is also beneficial for the backbone because the control traffic can be spread over different gateways and the single point of failure of the network can be removed. Additionally, if the backbone reliability is high and an adequate transmission mechanism is used to share local information among gateways in the East and Westbound APIs, the overhead in the backbone can be equal to the CCP operation mode.

6.3 Proof of concept: reactive congestion detection and node reconfiguration

6.3.1 Setup. In order to showcase the ability of the SDN-BLE framework to detect and automatically solve a network congestion (something that cannot be done in the current official BLE mesh protocol), a 12 node mesh network is set up, with default $CI = 300$ ms, $P_{TX} = 0$ dBm, $B = 25$ packets, $D = 1$, a data packet generation period of $T_{DP} = 10$ s and a control packet generation period of $T_{CP} = 30$ s for each node. This scenario is introduced in [26].

Figure 13 shows the topology of the straight-line network, where every node is only allowed to have a master and a slave connection. Albeit simple, this topology represents the worst case scenario in terms of congestion: as each node generates own data packets and routes them towards the sink node N_{12} , nodes closer to the sink need to deliver a higher load of traffic [14]. The data rate is constant for all nodes in the network with the exception of N_1 , which eventually starts generating 2 data packets per second ($T_{DP} = 0.5$ s, 20 times lower than the rest of nodes). As introduced in Section 4.3.2, this will translate into all intermediate links in the network being congested, so the SDN-BLE framework will need to detect such congestion and determine that the CI of all nodes needs to be decreased so the capacity of the network is enough to support traffic demands.

Fig. 13. Single line multihop data mesh network. Each node transmits its own data and forwards packets from the previous nodes at a constant rate, except N_1 which has a variable packet generation rate.

Fig. 14. (a) Elapsed average buffer occupancy for all nodes (μ_B) and data plane throughput (R_{DP}). (b) Elapsed end-to-end data packet delay (Δt_{DP}) and cumulative data packet losses.

6.3.2 Network states and Proof of Concept results. When N_1 abruptly increases its packet generation rate the network transitions from a **Steady State** into a **High Demand State**, where the CI of the nodes cannot support this data rate. Figure 14a depicts the buffer occupancy of the network averaged for all nodes (μ_B), elapsed for every received data packet at the sink from N_1 . Since $CI = 300$ ms is not enough to deliver packets at this rate, the buffers from all intermediate nodes in the path start to populate, until eventually all buffers are full and the network becomes **Congested**.

Once congested, the Anomaly Detector sees a decrease in each node R_{DP} and eventually obtains that $\mathbb{P}_A = 1$ for all nodes, so an anomaly report is sent to the Dynamic Node controller. The Dynamic Node Controller then obtains that the buffer occupancy is increased and therefore determines that the network is congested and from the Network Topology Builder extracts that all nodes should be reconfigured as it is a single-line topology. Then, it generates a Southbound API downlink packet to decrease the CI of all nodes (avoiding the congested links) so the high demand can be supported. When the nodes receive such packet they reconfigure their CI to the lowest possible value ($CI = 10$ ms in the framework) and the network is able to recover and return to the **Regular Performance** state.

Figure 14a additionally depicts the average throughput of the network R_{DP} . In the **Steady State** the average throughput is $R_{DP} = 176$ bps, since all nodes transmit a data packet every 10 seconds. In the **Congested State** the throughput increases to $R_{DP} = 5.8$ Kbps, where the nodes transmit at their maximum possible rate for their CI , which is not enough to satisfy network demand. This is only achieved once the anomaly is solved, resulting in a throughput of $R_{DP} = 9$ Kbps when the final **Regular Performance** state is reached.

Figure 14b depicts the end-to-end data packet delay Δt_{DP} and the cumulative data packet loss elapsed for every received packet. The packet delay ramps up in the **High demand** state when the buffers start to populate and saturates to its maximum value in the **Congestion** state, where every

packet that is not dropped due to buffer overflow travels through all buffers from all nodes in the network, resulting in $\Delta t_{DP} = 75$ s. Note that this curve is not aligned with the four states marked precisely due to the delay of the received packets. The delay decreases back to its original value once the anomaly is solved and the **Regular performance** state is reached. Regarding the cumulative data packet loss, the high P_{TX} and ACKs and retransmissions used ensure that packets are only lost due to buffer overflow. This only occurs in the **Congestion state**, and once the congestion is solved this values does not increase anymore.

6.3.3 Conclusion. The proposed SDN-BLE mesh framework is able to reactively detect network congestion and the nodes causing this situation. Then, it is able to individually reconfigure over the air the parameters of the nodes involved using the dedicated control plane information flow and solve the congestion without interfering with the application data information flow. Without the SDN-BLE mesh infrastructure in place, this congestion situation would not have been detected nor solved. In the hospital scenario considered, this would render the patients' sensors completely useless as no measurements could be sent to the central server.

For this proof of concept, $D = 1$ and $T_{CP} = 30$ s to showcase the utility of the framework, and an exploration of the effect of these parameters on the delay in which the system is able to detect and solve the anomaly is left as future work. In any case, decreasing D will decrease the reliability of the control plane information flow while reducing the energy consumption, while increasing the rate of control packets can have a negligible effect in network overhead while reducing the time needed for the CP to automatically detect the congestion and solve it.

7 CONCLUSIONS

This work elaborates on the implementation challenges of SDN for WSN and IoT and motivates BLE mesh as a suitable technology for an all wireless SDN-BLE solution, as it can solve the two main challenges of wireless SDN in terms of isolation of control and data plane information flows and reliability and scalability of the control plane.

Following the classical SDN paradigm, the design and implementation of all the different layers of the SDN-BLE framework is given. The proposed hybrid BLE mesh architecture, where the data plane information flow is assigned to a FruityMesh based network routing slice and the control plane information flow is assigned to a Trickle based flooding network slice ensures isolation of paths, channels, timestamps, buffers and state machines. The ability to independently fine tune the corresponding parameters of these two network slices offers the possibility to adapt network behavior and reliability to satisfy application requirements. Moreover, the design of the duty cycle approach is able to reduce the additional energy consumption introduced by the control plane activity. Additionally, centralized and distributed control plane operation modes are described, where a distributed approach over several gateways in a WiFi backbone can remove the single point of failure of the network that the centralized controller entitles, which is classically the main reliability concern of SDN.

An experimental characterization of the SDN-BLE framework is given. The effects of the duty cycle and the network size in the control plane reliability are measured, where the results show that these two factors can be traded to obtain a certain reliability, where increasing the duty cycle translates into higher energy consumption while increasing the network size translates into higher network overhead. Additionally, the estimated battery lifetime as a function of the duty cycle and the control plane activity rate is obtained, where the results show that the main factor affecting it is the former, and the contribution of the latter can be neglected.

Regarding the distributed versus centralized control plane operation mode discussion, the experiments conducted conclude that a distributed mode is beneficial for the BLE mesh when the

network size is large, which is in turn the desired scenario in which a distributed approach can ease network management.

Finally, a proof of concept for the SDN-BLE network is presented, where a network congestion can be detected reactively detected and automatically solved, a task that would not be possible without the infrastructure in place and that is impossible to achieve in current WSNs.

A main conclusion obtained is that the performance of the system depends on multiple parameters that may have opposing effects in the different metrics used for its characterization, and in general many trade-offs need to be taken into account. With this in mind, a discussion on the optimal performance of the system does not bring any concrete outcome, and an optimality in the Pareto sense is more adequate, where high network reliability, low network overhead, low end-to-end latency and low energy consumption are the four main targets of the framework, in which at best only three of them can be achieved in any configuration.

ACKNOWLEDGMENTS

Y. Murillo is funded by a Ph.D. fellowship of the Research Foundation Flanders (FWO). A. Chiumento is funded from the European Union's Horizon 2020 research programme under the Marie Skłodowska-Curie grant agreement No 713567 and this material is based upon work supported by the Air force Office of Scientific Research under award number FA9550-18-1-0214 and supported by the Science Foundation Ireland under Grant No. 13/RC/2077.

REFERENCES

- [1] Michael Baddeley, Reza Nejabati, George Oikonomou, Mahesh Sooriyabandara, and Dimitra Simeonidou. 2018. Evolving SDN for Low-Power IoT Networks. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. 71–79. <https://doi.org/10.1109/NETSOFT.2018.8460125>
- [2] Samaresh Bera, Sudip Misra, and Athanasios V Vasilakos. 2017. Software-Defined Networking for Internet of Things: A Survey. *IEEE Internet of Things Journal* 4, 6 (Dec 2017), 1994–2008. <https://doi.org/10.1109/JIOT.2017.2746186>
- [3] Bluetooth SIG. 2019. Bluetooth Core Specification v5.1. Kirkland, WA, USA.
- [4] Bluetooth SIG. 2019. Mesh Profile Specification v1.0.1. Kirkland, WA, USA.
- [5] Alessandro Chiumento, Brecht Reynders, Yuri Murillo, and Sofie Pollin. 2018. Building a connected BLE mesh: A network inference study. In *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 296–301. <https://doi.org/10.1109/WCNCW.2018.8369004>
- [6] Keuchul Cho, Woojin Park, Moonki Hong, Gisu Park, Wooseong Cho, Jihoon Seo, and Kijun Han. 2014. Analysis of Latency Performance of Bluetooth Low Energy (BLE) Networks. *Sensors (Basel, Switzerland)* 15 (12 2014), 59–78. <https://doi.org/10.3390/s150100059>
- [7] Mario Collotta and Giovanni Pau. 2015. A Novel Energy Management Approach for Smart Homes Using Bluetooth Low Energy. *IEEE Journal on Selected Areas in Communications* 33, 12 (Dec 2015), 2988–2996. <https://doi.org/10.1109/JSAC.2015.2481203>
- [8] Nick Feamster, Jennifer Rexford, and Ellen Zegura. 2014. The Road to SDN: An Intellectual History of Programmable Networks. *SIGCOMM Comput. Commun. Rev.* 44, 2 (April 2014), 87–98. <https://doi.org/10.1145/2602204.2602219>
- [9] Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh. 2011. Efficient network flooding and time synchronization with glossy. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*. IEEE, 73–84.
- [10] Carles Gomez, Ilker Demirkol, and Josep Paradells. 2011. Modeling the Maximum Throughput of Bluetooth Low Energy in an Error-Prone Link. *IEEE Communications Letters* 15, 11 (November 2011), 1187–1189. <https://doi.org/10.1109/LCOMM.2011.092011.111314>
- [11] Matteo Grandi, Daniel Camps-Mur, August Betzler, Joan Josep Aleixendri, and Miguel Catalan-Cid. 2018. SWAM: SDN-based Wi-Fi Small Cells with Joint Access-Backhaul and Multi-Tenant Capabilities. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 1–2.
- [12] Hardkernel. [n.d.]. Odroid-C2 single-board computer website.
- [13] Moeen Hassanalieregh, Alex Page, Tolga Soyata, Gaurav Sharma, Mehmet Aktas, Gonzalo Mateos, Burak Kantarci, and Silvana Andreescu. 2015. Health Monitoring and Management Using Internet-of-Things (IoT) Sensing with Cloud-Based Processing: Opportunities and Challenges. In *2015 IEEE International Conference on Services Computing*. 285–292. <https://doi.org/10.1109/SCC.2015.47>

- [14] Jangeun Jun and Mihail L Sichitiu. 2003. The nominal capacity of wireless mesh networks. *IEEE wireless communications* 10, 5 (2003), 8–14.
- [15] R Kelsey and J Hui. 2016. Multicast protocol for low-power and lossy networks (MPL). *IETF, RFC 7731* (2016).
- [16] Hamidreza Kermajani and Carles Gomez. 2014. On the network convergence process in RPL over IEEE 802.15. 4 multihop networks: Improvement and trade-offs. *Sensors* 14, 7 (2014), 11993–12022.
- [17] Hlabishi I. Kobo, Adnan M. Abu-Mahfouz, and Gerhard P. Hancke. 2017. A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements. *IEEE Access* 5 (2017), 1872–1899. <https://doi.org/10.1109/ACCESS.2017.2666200>
- [18] Diego Kreutz, Fernando Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. 2015. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* 103, 1 (Jan 2015), 14–76. <https://doi.org/10.1109/JPROC.2014.2371999>
- [19] P Lewis, T Clausen, J Hui, O Gnawali, and J Ko. 2011. *The Trickle Algorithm*. RFC 6206. <https://tools.ietf.org/html/rfc6206>
- [20] Yunhao Liu, Kebin Liu, and Mo Li. 2010. Passive Diagnosis for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking* 18, 4 (Aug 2010), 1132–1144. <https://doi.org/10.1109/TNET.2009.2037497>
- [21] Tie Luo, Hwee-Pink Tan, and Tony Q.S. Quek. 2012. Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks. *IEEE Communications Letters* 16, 11 (November 2012), 1896–1899. <https://doi.org/10.1109/LCOMM.2012.092812.121712>
- [22] M-Way Solutions. [n.d.]. FruityMesh. <https://github.com/mwaylabs/fruitymesh>
- [23] Marcio L. F. Miguel, Edgard Jamhour, Marcelo E. Pellenz, and Manoel C. Penna. 2018. SDN Architecture for 6LoWPAN Wireless Sensor Networks. *Sensors* 18, 11 (2018). <https://doi.org/10.3390/s18113738>
- [24] Mikrotik. [n.d.]. RB912UAG-2HPnD routerboard website. <https://mikrotik.com/product/RB912UAG-2HPnD>
- [25] Esteban Municio, Johann Marquez-Barja, Steven Latr, and Stefano Vissicchio. 2018. Whisper: Programmable and Flexible Control on Industrial IoT Networks. *Sensors* 18 (11 2018), 4048. <https://doi.org/10.3390/s18114048>
- [26] Yuri Murillo, Alessandro Chiumento, Brecht Reynders, and Sofie Pollin. 2019. SDN on BLE: Controlling Resource Constrained Mesh Networks. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 1–7.
- [27] Yuri Murillo, Brecht Reynders, Alessandro Chiumento, Salman Malik, Peter Crombez, and Sofie Pollin. 2017. Bluetooth now or low energy: Should BLE mesh become a flooding or connection oriented network?. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. 1–6. <https://doi.org/10.1109/PIMRC.2017.8292705>
- [28] Yuri Murillo, Brecht Reynders, Alessandro Chiumento, and Sofie Pollin. 2019. A Multiprotocol Low-Cost Automated Testbed for BLE Mesh. *IEEE Communications Magazine* 57, 3 (March 2019), 76–83. <https://doi.org/10.1109/MCOM.2019.1800122>
- [29] Clara Nieto Taladriz, Yuri Murillo, and Sofie Pollin. 2018. Towards Efficient BLE Mesh: Design of an Autonomous Network Joining Algorithm. *2018 IEEE Symposium on Communications and Vehicular Technology (SCVT'18)*.
- [30] Nordic Semiconductor. [n.d.]. Nordic NRF52 development board website. <https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52-DK>
- [31] Nordic Semiconductor. [n.d.]. Nordic NRF52 online power profiler. <https://devzone.nordicsemi.com/power/>
- [32] Open Networking Foundation. 2012. Software-Defined Networking: The New Norm for Networks. (04 2012). <https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf>
- [33] Matti Siekkinen, Markus Hienkari, Jukka K. Nurminen, and Johanna Nieminen. 2012. How low energy is bluetooth low energy? Comparative measurements with ZigBee/802.15.4. In *2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 232–237. <https://doi.org/10.1109/WCNCW.2012.6215496>
- [34] Milosh Stolikj, Thomas M. M. Meyfroyt, Pieter J. L. Cuijpers, and Johan J. Lukkien. 2015. Improving the Performance of Trickle-Based Data Dissemination in Low-Power Networks. In *Wireless Sensor Networks*, Tarek Abdelzaher, Nuno Pereira, and Eduardo Tovar (Eds.). Springer International Publishing, Cham, 186–201.
- [35] Pascal Thubert. 2018. *An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4*. Internet-Draft draft-ietf-6tisch-architecture-19. Internet Engineering Task Force. <https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-architecture-19>. Work in Progress.
- [36] Mostafa Uddin, Sarit Mukherjee, Hyunseok Chang, and TV Lakshman. 2017. BLESS: Bluetooth low energy service switching using SDN. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 571–576. <https://doi.org/10.1109/INFCOMW.2017.8116440>
- [37] Philipp Zenker, Silvia Krug, Michael Binhack, and Jochen Seitz. 2016. Evaluation of BLE Mesh capabilities: A case study based on CSRMESH. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 790–795.
- [38] Anatolij Zubow, Michael Dring, Mikolaj Chwalisz, and Adam Wolisz. 2015. A SDN approach to spectrum brokerage in infrastructure-based Cognitive Radio networks. In *2015 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 375–384.