Demo Abstract: BE-Mesh: Bluetooth Low Energy Mesh Networking

Andrea Lacava*, Gianluigi Nero *, Pierluigi Locatelli *, Francesca Cuomo*, Tommaso Melodia†
* University of Rome "La Sapienza", 00184, ITALY † Northeastern University, MA 02115, USA

***Abstract*—We propose and discuss *BE-Mesh-Bluetooth low Energy-Meshed network*, a new paradigm for BLE (Bluetooth Low Energy) that enables mesh networking among wirelessly interconnected devices, both in a single hop and multi-hop fashion. Starting from the classical Master/Slave paradigm of Bluetooth, we build two new layers based on BLE stack that allow the final user to set-up, in a fast way, the desired network topology while hiding the complexity and low-level details of the BLE stack. We also prototype, as a proof of concept, an open source Android library [1] that implements our communication paradigm and an Android application that allows the exchange of text messages across the mesh network. Last, we demonstrate how BE-Mesh enables Internet access sharing with the whole mesh from a single Internet-connected device.**

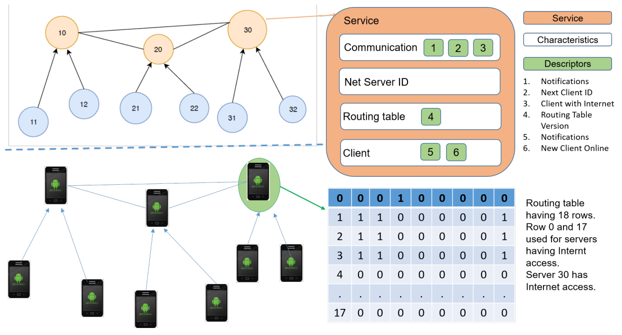
I. Introduction

The Bluetooth Low Energy (BLE) short range wireless technology is rapidly expanding and represents the frontier for device to device smartphone communications and IoT development [2][3]. Most existing BLE networks use a single controller device that manages all interactions with peripheral devices. This has, so far, limited the potential of smartphones to manage complex networks and has slowed the development of fields such as home automation and IoT. We notice that the scatternet formation (i.e. the bridging of different piconets) has been a complex problem, studied for over a decade, and not fully solved till now [4]).

In this demo, we present a new paradigm of communication based on classical BLE stack which allows the creation of a smartphone and IoT network based on Multiple Advertisements [2] and we provide an Android library [1] as a proof of concept. In the current BLE literature [5][6] there is not a similar solution of communication for smartphones that allows Android phones (of any kind of OS and Bluetooth chip) to dynamically set up, in a distributed way, a mesh network used to communicate with each other without a permanent pairing. Moreover, our formed BE-Mesh network is always-on with no need of continuous switching of nodes acting as bridges between different piconets.

The native features of BLE support a fast information exchange with low energy consumption. We identify two user cases where our BE-Mesh can be applied:

- BE-Mesh is suitable to be used in emergency scenarios where long-distance communication (cellular and wired terrestrial networks) may no longer be available.
- BE-Mesh can be used in the home automation field: with our network we can interconnect all the home smart devices and give full Internet connection to all of them by having a single device with Internet access that can share with the others, via BE-Mesh.



0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	1
2	1	1	0	0	0	0	0	1
3	1	1	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0
.	.	.	.	.	.	.	.	.
17	0	0	0	0	0	0	0	0

Fig. 1: Example of topology with Server's service description and a BE-Mesh client routing table; the client in green is the unique having direct Internet access

II. High level interface and Communication

We define the high level of BE-Mesh as the collection of roles and utilities exposed to library users. At this level we have two main entities representing the operating device in the network: the *client* and the *server*.

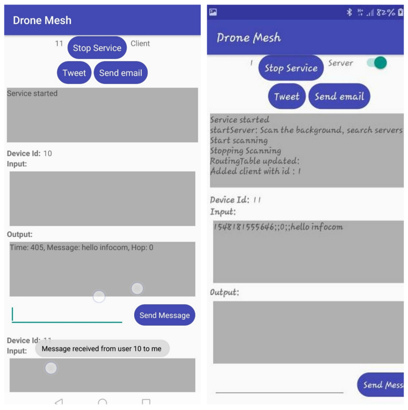
The client: i) has a list of the other devices active in the network; ii) can send and receive messages with other devices; iii) it is not paired permanently with a server (differently from the standard Bluetooth that has a persistent connection). Data is publicly advertised via BLE without any need of a pre-established pairing, so a client is linked to just one server and, as a consequence, iv) his next hop is always a server. The server is responsible for the communication and data exchange so it needs to support both BLE and Multiple Advertisement. It: i) stores and updates a routing table representing the whole network, ii) has the characteristics and the descriptors to exchange data, iii) is linked with the other servers.

Every device in BE-Mesh has its own identifier. Servers have an ID $i \in \{1, 16\}$, while clients have an ID ij where i is the number of the associated server and j is a number $\in \{1, 7\}$. The distinction between client and server is necessary to force devices that do not support Multiple Advertisement [2] or that do not have basic functions (e.g., an IoT device) not to be a central node for the network; this guarantee the versatility of BE-Mesh network.

At the startup, a device scans the surrounding environment to verify if a preexisting network exists and if a server is

TABLE I: Data structure Server side

Service	Description
Communication	Used for clients messages exchange and server-level communication. This characteristic has 3 descriptors: (i) *Notification descriptor*: used at the client-side to subscribe to this characteristic in order to be notified whenever the value changes, making it possible for the client to receive messages. (ii) *Next Id descriptor*: it contains the value of the next ID to be assigned to a new client (maximum 7). (iii) *Client With Internet descriptor*: here a client writes its own ID to indicate to the server it has access to Internet.
Routing table	Used to send and receive the routing table, used only at the server-side to globally synchronize the routing table. It has a *Version descriptor* that contains the version number of the routing table, that makes servers understand when it is needed an update of the table.
Next Server ID	A globally synchronized value that represents the next server ID available in the BE-Mesh network.
Client Online	Used by the server to keep a list of active clients. It has two descriptors, the (i) *Notification descriptor*: used like the communication one to receive the new list of clients online and the (ii) *New Client Online descriptor*: used by servers to communicate whenever a new client is online.



Service started

Time: 405, Message: hello infocom, Hop: 0

Message received from user 10 to me

Service started
startServer: Scan the background, search servers
Start scanning
Stopping Scanning
RoutingTable updated:
Added client with id : 1

1548181555646;;0;;hello infocom

Fig. 2: Screen snapshot of a client receiving a message from a server. Client side on the left, server on the right.

available; in the latter case the device tries to become its client. If there are no servers available, the device becomes a server and selects its unique ID. We denote as *Scan Collision Problem* when collisions of IDs happen due to simultaneous scanning by two or more devices that could became servers with the same unique ID. To solve this problem we implemented a pseudo-random timing solution: if no servers are found, the device goes into an idle state for a pseudo-random time, arbitrarily set. At the end of this time a new scanning is performed and if again there are no servers, the waiting time is repeated again till a maximum value. If at the end of this scanning the device does not found any server it assumes that there are no existing networks and if it has all the requested characteristics it becomes the server #1. This process is also repeated in case of server's crash. After fault detection clients clean their data structures and then start all over.

III. Low level interface and Communication

Communication in the lower level is based on data structures present in servers, described in Figure 1 and in Table I.

At the startup, a device performs different operations based on the assigned role: a server initializes its service and data to their default values and a client tries to collect his unique ID contained in the server's *Next ID descriptor*. After the assignment of the ID, the client has to enable the notification on two characteristics: (i) *Communication Characteristic* to receive messages and (ii) *Client Online Characteristic* to know when another client is online. After these operations the client is ready to take part to the network and it reads the *Client Online* characteristic in order to start exchanging data with other clients in the network. Finally, there are four types of communications users in the network: (i) normal messages between nodes/internet messages; (ii) routing table updates between server; (iii) new server in the network; (iv) new client in the network.

IV. Demo

In the demo we show an implementation of BE-Mesh in Android [1] and we demonstrate its functionality in the multi-hop (scatternet based) configuration. We will show the whole network start-up procedure presented in this paper, especially the possibility for every device to send an email or to tweet through BE-Mesh. These messages will be first delivered in the network from an offline device to a device with an Internet access and then sent on the Internet by using standard third parts APIs. Every participant will be invited to download our app from the Play Store [7]. We will ask to all of them, apart one device, to close their mobile traffic data. Then everyone would start the demo simply by clicking the start button. Once the whole network will be configured, users will be able to exchange messages. The fact that there are users with Internet enabled and some users without a connection that can still send emails and tweets allow us to demonstrate the enormous potential of BE-Mesh to provide the Internet to any type of smartphone or IoT device regardless of its primary purpose. A short video preview of our demo is available at [8].

Acknowledgement

The authors would like to thank Ludovico Ferranti for the support given in the application design. The work of Prof. Melodia was supported in part by the US National Science Foundation under grant NSF CNS-1618727.

References

[1] https://github.com/Thecave3/drone-ble-mesh/tree/master/, 2018, [Online; accessed 23-January-2019].

[2] S. Bluetooth, "Bluetooth core specification version 4.0," *Specification of the Bluetooth System*, 2010.

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

[4] F. Cuomo, T. Melodia, and I. F. Akyildiz, "Distributed self-healing and variable topology optimization algorithms for qos provisioning in scatternets," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 7, pp. 1220–1236, 2004.

[5] L. Leonardi, G. Patti, and L. L. Bello, "Multi-hop real-time communications over bluetooth low energy industrial wireless mesh networks," *IEEE Access*, vol. 6, pp. 26 505–26 519, 2018.

[6] S. Darroudi and C. Gomez, "Bluetooth low energy mesh networks: A survey," *Sensors*, vol. 17, 06 2017.

[7] https://play.google.com/store/apps/details?id=it.drone.mesh, 2019, [Online; accessed 23-January-2019].

[8] G. Nero, P. Locatelli, and A. Lacava, https://youtu.be/sKnjs0BUny4, 2019, [Online; accessed 23-January-2019].