Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 711–720
DOI: 10.15439/2023F287
ISSN 2300-5963 ACSIS, Vol. 35

Semi-persistent services for IoT networks using RESTful approach

Jarogniew Rykowski
0000-0001-7944-6061
University of Economics and Business, Department of Information Technology
Niepodleglosci 10, 61-875 Poznan, Poland
Email: rykowski@kti.ue.poznan.pl

Abstract— **The paper proposes a new way to access semi-permanent services in ad-hoc and mesh networking in the context of the Internet of Things and the Internet of Services. The solution is based on address-free communication, with individual addresses of the nodes replaced by a semantic description of their functionality. The mesh network is accessible from outside using the classic RESTful approach and needs no centralized catalog to maintain at-the-moment available services. Instead, entry gateways are responsible for mapping incoming REST-compliant communication to internal mesh messaging, and the mesh nodes individually decide how to react to particular messages. Installing each new or replaced node or monitoring the node status is unnecessary. Automatic communication among the nodes is possible without human intervention, including both runtime and the registration phase.**

Bluetooth mesh topology network was chosen as the implementation base. Transmission in the network occurs in a broadcast mode, in which one network node sends information that is then received and interpreted by all other nodes. Selected devices equipped with alternative communication modules of a different type, in particular, connected to the home WiFi network, can be used as input/output gateways for outside communication.

Index Terms—**address-free networking, zero-configuration networking, P2P broadcasting networks, BLE mesh, ad-hoc networks, RESTful approach.**

I. Introduction

Recently we have observed a boom in the client-server architecture. The RESTful approach needs special attention among multiple protocols and strategies used to implement a client-server system [1, 2]. This is an architectural style based on standard HTTP requests, and URL addressing. The key entity for a REST-compliant system is the so-called ‘resource’. A REST resource is a well-defined part of server functionality, being a building block for the set of servers’ services. A resource could be a pointer to a file, a server-side script, a database entry, etc. Each resource is uniquely identified by means of an URL address [3], composed of the scheme identifier, domain name, port number, a local path to a file/script, and a set of parameters composing a query.

The RESTful approach is an efficient tool for implementing a fixed, well-defined network. On the contrary, recently, we have observed the growing importance of ad-hoc [4] and dynamic networking, especially in the context of vehicular and road networks [5], not to say about classical sensor networks [6] and MANETs [7, 8]. The research in the domain of ad-hoc networking recently concentrated on mobility, energy savings, and routing [9], especially under specific circumstances (such as underwater acoustic connections [10], also evolving into an interesting concept of a digital twin [11]. Recently, a new approach to distributing network functionality, namely the fog [12], has become popular, concentrating some resources (nodes, computation power, memory, etc.) close to the most needed places in order to provide real-time reaction.

For ad-hoc networking, nothing is fixed – neither the set of services/nodes nor the information contents. Especially, ad-hoc networking strongly limits fixed ways of addressing network services (resources) by their location and name, leading to the concept of address-free networking [13], which is well known but somehow abandoned recently. Thus, using classical URLs in the ad-hoc system is disputable unless the server-side is fixed and only the clients are connected incidentally. As a consequence, server-side architecture for ad-hoc applications is usually replaced by a mesh. A mesh network [14] is a local area network topology in which the nodes connect directly, dynamically, and non-hierarchically with as many other nodes as possible and cooperate with one another to route data efficiently. A mesh node is not characterized by a permanent identification (e.g., an address); moreover, such a node may suddenly vanish from the network structure or convert to another node due to network evolution, such as the one provoked by the mobility of nodes, amount of available energy, problems with radio transmission, etc. A popular example of a mesh network is BLE (Bluetooth Low Energy) Mesh proposal [15, 16], with several implementations [17] and sub-standards/extensions [18], also related to security and privacy [19], and interoperability [20], addressed to not only classical computers, tablets, or smartphones, but also IoT (Internet of Things) devices based on (among others) Espressif, STM, and Nordic Semiconductor processors with built-in BLE unit.

A question arises if it is possible to apply the RESTful approach with its persistent services, which is very efficient and thus popular in fixed/stable networking, to mesh networking? At first view, the answer is ‘no’. Indeed, if we cannot identify a node for a longer time, we cannot establish a client-server connection to ask for a service. However, the above question may be converted to another one: is it possible to address REST-compliant persistent resources in a dynamic way, not using fixed URLs but mapping external REST calls to internal mesh messages of particular format and semantics? Thus, persistent addressing of the resources is to be replaced by a persistent (i.e., stable) description of their functionality. Later on, this fixed description may be used for individual selection of the nodes fulfilling certain criteria. The selection may be performed by the nodes – each node, based on its own declared functionality, accepts or refuses the incoming requests.

In the paper, we propose such an extension to provide RESTful information exchange for a mesh network, especially for BLE Mesh. The idea is to treat a mesh network as a set of REST resources of a given functionality, to be identified not by their URL addresses but by means of their characteristics and possibilities. From the outside, the mesh network is seen as a REST server with specific functionality. Internally, the nodes composing the dynamic network react individually to fulfill incoming REST requests. The node’s reaction depends on the individual characteristic of this node, in particular, its type being a counterpart of the REST resource name and the REST query.

The remainder of the paper is organized as follows. Section 2 overviews the motivation for using semi-persistent services in ad-hoc interactions. Section 3 describes a generic approach to mapping REST calls to mesh functionality and the reasons for particular network topology and organization. Sections 4 and 5 describe a sample implementation of kitchenware equipment based on our approach. Next, we include a comparison with similar work, and finally, we provide some conclusions and directions for future work.

II. Specificity of Mesh and Ad-hoc Networking

If we speak about ad-hoc networking, at first view, we think that every activity is undertaken incidentally: ad-hoc place and time, situation, context, etc. However, that is not the whole truth. There is one element that is fixed – the user. We do not change our needs and expectations just because we are, by coincidence, in an unknown situation. On the contrary, we try to act in a “usual” way, according to our past experience and customs. One may say that we stay with our needs, fixed, as long as it is possible, and we try to act “as usual” even if the case is extraordinary.

Thus, an idea arose to propose a new sort of services for an ad-hoc environment. The goal is to use a service in the same (or at least very similar) way, regardless of place and time. Such services are semi-persistent. In such a way, from the point of view of a user, they are the same everywhere, appearing for this user in the form the user expects. However, from the technical point of view, the services are independent entities, implemented individually and possibly adjusted to the place (conditions) where they work.

In an ad-hoc environment, the users do not know the specificity of the place they are currently in, particularly the set of nodes and their identifiers (addresses, services, and their entry points). However, they know the place's overall character (such as a home, a shop, a bus stop, etc.), and they expect some well-known services accessible at this place. For example, they expect hot water to be prepared in the kitchen or a bus going to a specific destination at the bus stop. Please note that location-specific services are usually well-defined and common for all places of the same type and purpose.

With a classical approach, such as a typical REST application, getting consistent services at many unrelated and unsynchronized places is almost impossible. For example, one cannot expect the same IP address and naming convention for network nodes. Moreover, an installation phase is needed to create some “entry points” for the services, such as installation and configuration of a specific application, catalog of available services and their status, etc. It is unrealistic to expect each place to follow the same rules of addressing and parameterizing, not to say about the security (user identification, access codes, passwords, etc.). As a consequence, we usually deal with one fixed application per place.

This paper proposes a different approach to preparing and accessing such semi-persistent services. We assume that a definition of the semantics of the services is fixed and shared by all the ad-hoc accessible places. Such a semantic description is also known for end-user devices, usually smartphones. Each time a user is at an unknown, ad-hoc place/situation, the description is used to formulate a request, to be disseminated across all the ad-hoc network nodes. If a node (or a set of nodes) “understands” the request, then this node undertakes particular action related to the semantic description, trying to fulfill the request. The network is “silent,” and no activity is performed if no single node can provide the service.

Please note that the node should individually choose the “implementation” of the requested service. Only the node knows in detail the specificity of the place, which is unknown to the user. So, the user may only formulate a generic request (such as “turn on some light here”), but how the request would be served (such as “switch on ceiling lamps to 50%”) depends on the possibilities and strategy of the place.

As it may be seen, the idea of accessing semi-persistent service in an ad-hoc manner is the following: “try it, and if you are lucky, the service is there for you; otherwise, try a different way or give up”. Such experiments are to be undertaken at any unknown (not previously visited, or changed for some reasons) ad-hoc place; however, they are quite natural and intuitive for humans. Moreover, these experiments somehow bypass the installation phase and need no a’priori catalog of at-the-place services.

Initially, we planned to implement our approach as a home application, namely, a "smart" kitchen. Usually, the kitchenware is (1) not synchronized, such as a kettle does not know about a presence of a radio, and an oven is not informed if a ventilator is here to reduce the smell while cooking, and (2) fixed as for the overall functionality (such as “a kettle” or “a refrigerator”), not necessarily fixed as for models/producers/functionality of specific devices. Even if all the “smart” devices are accessible via the same kind of network (which is, in our case – classic Bluetooth, and recently BLE Mesh), these devices are to be registered in a catalog, and in most cases, the smartphone applications are specific for a given model of a device. Each time a device is broken and changed, the user must update the catalog information or install a different application. Keeping with a single application and ad-hoc access to semi-permanent services would solve these problems, on the condition that the services describe the possibilities of typical devices (in our case – the set of devices of a single producer). As a consequence, each application

- will be useful at any “smart” kitchen, no matter its location,
- no additional security checks are needed (the users are granted to use the devices installed at the place they are currently visiting), and
- no installation and cataloging is needed, as well as no “user manuals” for different models of similar devices.

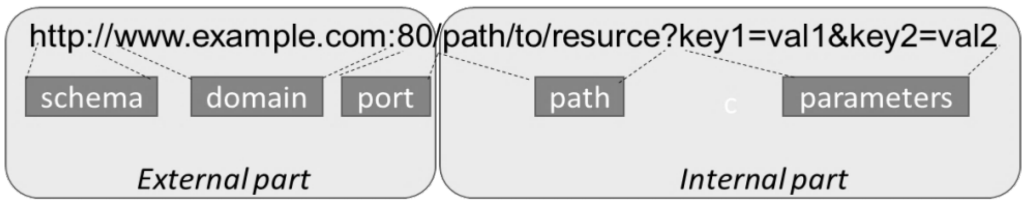
The semantic description of the at-the-place services covers all the details and frees users to learn detailed functionality (for some “smart” devices, quite complex, and, as previously mentioned, model-specific).

The proposed idea is generic and may be adjusted to many places and situations. The kitchenware application could be extended to any place, not necessarily private, but also public. For example, entering a bus stop, one can experiment with a service providing an actual timetable for the buses traveling to/from this bus stop in a minute. While visiting a shop, users may obtain additional information, e.g., the locations of the goods on the shelves, personalized advertisement, etc. At the school, the services may be related to the current schedule of the lectures (solving the “where is my next lecture” problem, etc.). The generic idea is that: entering an unknown place, the users ask for known services.

III. REST MODIFICATIONS TOWARDS THE USAGE IN MESH NETWORKS

As already mentioned, addressing a REST resource via HTTP calls aims in: providing computer identification (node address), declaring port number, defining a path to an internal entity implementing the REST resource (a file, a script, a database entry, etc.), and providing a query to adjust the resource’s behavior. Suppose we divide the above set into the “external” and “internal” parts. In that case, we may separate the node address and port number (“external” parameters), and the rest to be processed “internally” by the node. Further, we may logically link the “external” part with a gateway to the mesh network as a whole and the internal part with the given mesh functionality (to be, in turn, implemented by a set of mesh nodes). Such a global mapping is presented in Fig. 1.

As already mentioned, it is hard to identify a node in a mesh network permanently. Thus, linking REST resources with any node identifier is not justified. Instead, we propose to attach such a resource with certain well-defined, thus somehow persistent, functionality to be dynamically implemented by a node (or nodes). To clear the idea, we propose to join this functionality with a named type and to provide a map of resource names and their types. Types create a hierarchy, being a direct acyclic graph (DAG), precisely, a tree with a single root. Fig. 2 represents a sample hierarchy of types for typical kitchenware equipment we used for the sample implementation of the proposed approach.


http://www.example.com:80/path/to/resurce?key1=val1&key2=val2
schema
domain
port
path
parameters
External part
Internal part
Fig. 1. External and internal parts of URL address

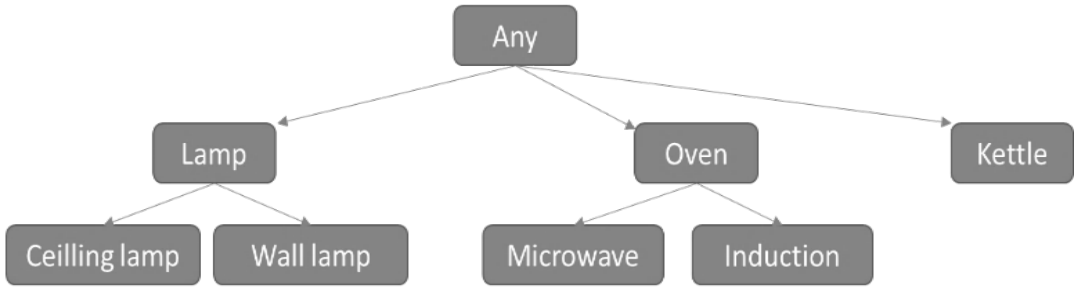


Fig. 2. Sample hierarchy of types of kitchenware equipment

As the types form a DAG hierarchy, they may be specialized or extended, depending on the point of view and the direction of graph analysis. For example, an 'oven' node may group "microwave oven", "gas oven", "induction oven" etc. Please note that all these "ovens" share some functionality, such as "set power to 1000W", and in turn have common functionality with some other electrical devices, such as "run timer". In particular, the root (a device of type "any") could possibly be linked with some generic functionality of any electric device, such as "on", "off", and the above "timer" actions.

According to the REST principle, we specialize some types, such as the above "ovens" share the basic functionality of an oven, and some specific functions of "oven/microwave", "oven/grill", "oven/induction", "oven/induction/plate_NE", etc. Please note that the above naming schema conforms to the REST principle.

The types may also be extended to individual names. For example, a "lamp" can be specialized to a "lamp/ceiling" and a "lamp/wall", sharing exactly the same functionality but differing in, e.g., the place of installation of the physical object the mesh node is connected to. In general, detailed organization and interpretation of the hierarchy depend on the application area and the installation place. As described later in the text, the hierarchy may be reflected by nodes' data, to be processed in a distributed manner – there is no need to represent the hierarchy as a whole by any network node or somewhere in a cloud. Furthermore, last but not least, addressing the primary REST name (such as the "lamp" in the above example) would activate all the devices of the same basic type. If "on" command is sent to all the "lamps", they will be all activated. However, if such a command is sent to "lamp/wall", only this lamp is illuminated.

Note that the above-described mapping shares the basic ideas of object-oriented programming, namely encapsulation, inheritance, and abstraction. Each mesh node may be treated as a singleton, with a well-defined set of "private" variables and functions and a "public" interface. This interface accepts only the "known" (from the node's point of view) incoming messages. As the nodes share the persistent inheritance hierarchy (c.f., Fig. 2), each node may check if the incoming message is of any of the "known" types. In such a way, the node must know both the type (mapped from the REST resource name) and the query (mapped from the REST parameters) to react; however, such a reaction is individual for each node.

As the mesh network structure is dynamic, and we do not provide such functionality as a centralized directory, one cannot apply direct identification of nodes. Thus, unicast transmissions are not possible. Instead, we propose to broadcast all the messages in the "non-responding" way. Each message comes to any node within the radio range. There, a preliminary checking occurs if such a message addresses a type that is linked with the node. If any node's type matches, the node interprets the message according to the query parameters. Appropriate action is undertaken (such as switching the real device linked with the node "on" or "off"). Thus, the nodes may react differently to the same messages. For example, an "alarm" message will close the windows, open the doors, switch on the lights, activate a siren, etc.

As it may be seen, a single broadcasted message may provoke quite complicated behavior of the mesh network, depending on the individual functionality of each network node. On the contrary, some messages are possibly not served once there is no node with certain functionality. For example, if an "oven" device asks for some ventilation, and no node with "a ventilator" type is provided, no air flow is initiated. If, however, at any time, an owner of the network decides to buy a new ventilator, this device will be activated with no changes in the existing network structure and nodes' functionality/hardware/software.

Broadcast messaging limits the way of possible response from the nodes. However, direct responding may be replaced by two succeeding broadcast transmissions. For example, if a device (e.g., a smartphone) is interested in temperature measurement, its node may broadcast a message "get temperature". Thus, each thermometer node reacts by broadcasting the "current temperature" parameterized in the query by the temperature value (Fig. 3). If no "response" broadcast message is observed for a certain period, this means there is no thermometer in the network. On the contrary, if several thermometers exist, they will all send a broadcast transmission with their values. It is up to the caller to get only the first (the highest, the lowest) one or to fetch them all and compute an average value.

IV. System architecture and data flow

We assume that a mesh network is equipped with at least one gateway to any public network (LAN or Internet). The node with the gateway has two network connections (Fig. 4): a public one (such as WiFi or LTE), and a mesh-related one (such as BLE Mesh). The gateway node is responsible for the mapping of incoming REST resources to internal mesh messages representing REST calls and eventually collecting the responses (i.e., reverse broadcast messages coming back within a certain timeout window) from mesh nodes to form an HTTP public response for the external call (c.f., "fridge" node from Fig. 3).

The mapping aims to process the URL address of a REST resource in several steps:

- cutting off the address part and port number,
- mapping resource name to type name,
- passing the incoming query with no changes,
- adding some specific query parameters, such as gateway identifier, the node number of the message sender, etc.,
- encrypting the message using BLE Mesh keys.

The mesh message is sent to any other node in the network (Fig. 5). If the network is big enough, so the relaying is needed, then some nodes may re-send the message to some other nodes. Standard BLE Mesh mechanism is used to relay the messages, and to eliminate incidental message copies in a reasonable time window (usually a second).

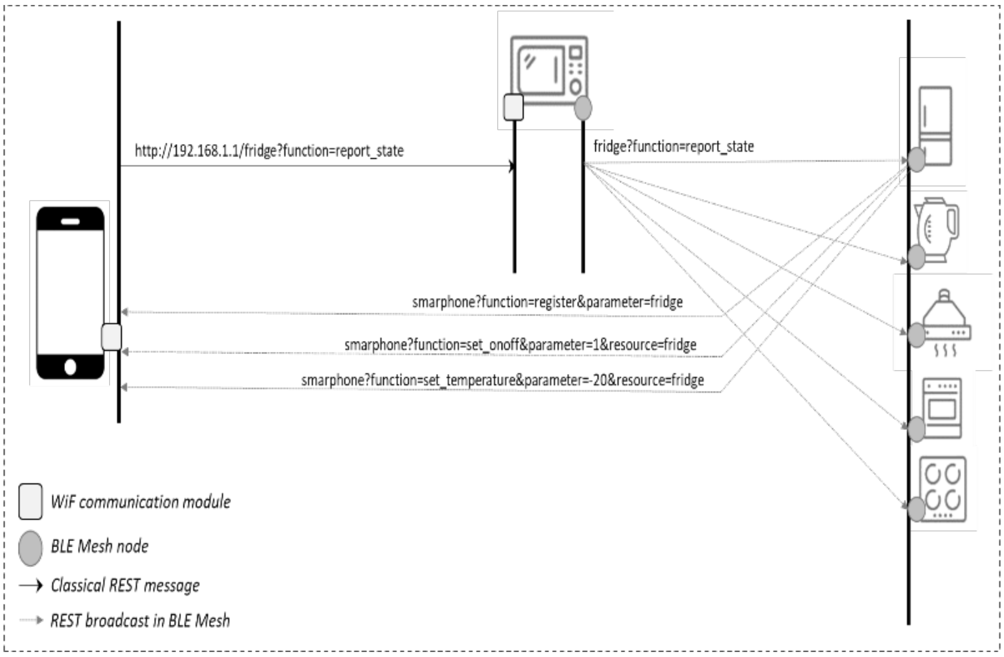

http://192.168.1.1/fridge?function=report_state
fridge?function=report_state
smarphone?function=register&parameter=fridge
smarphone?function=set_onoff&parameter=1&resource=fridge
smarphone?function=set_temperature&parameter=-20&resource=fridge
WiF communication module
BLE Mesh node
Classical REST message
REST broadcast in BLE Mesh

Fig. 3. Typical way of interaction via broadcast messaging

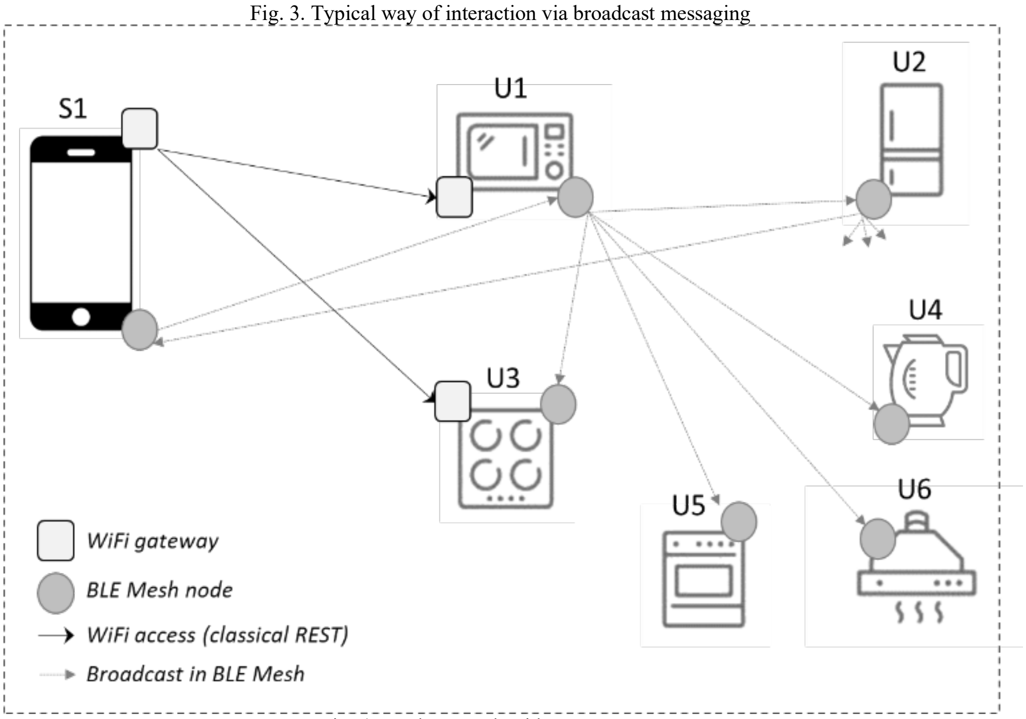


Fig. 4. Mesh network with entry gateways

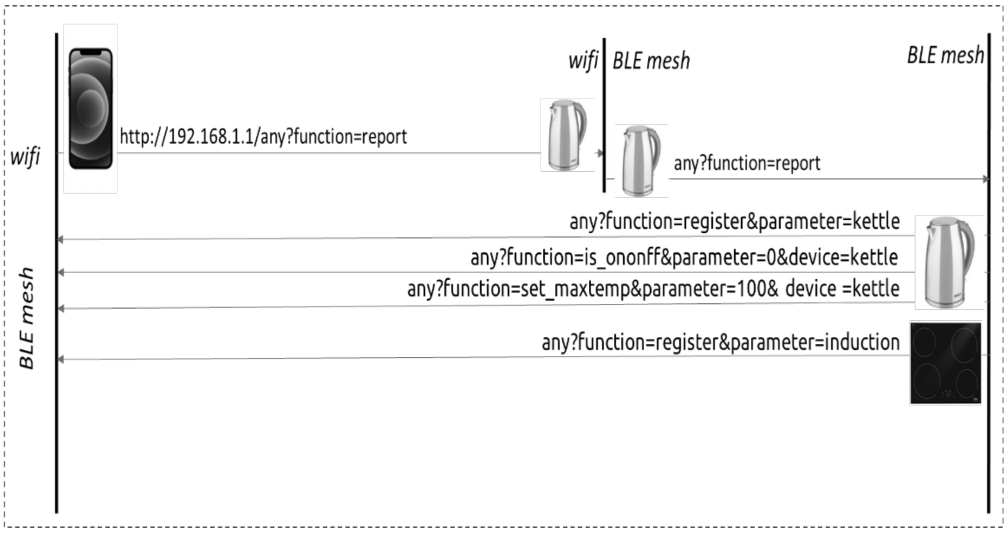

wifi
BLE mesh
http://192.168.1.1/any?function=report
wifi
BLE mesh
any?function=report
BLE mesh
any?function=register&parameter=kettle
any?function=is_ononff&parameter=0&device=kettle
any?function=set_maxtemp&parameter=100& device =kettle
any?function=register&parameter=induction

Fig. 5. Mapping messages in the gateway node

Each node, after receiving the mesh message, is responsible for (1) unfolding the non-standard characters using the standard *URLdecode()* function, (2) checking the accordance of the type, and (3) if the message type conforms to any type declared for the node, consume the message and start appropriate action(s) addressed to the real-world entity the node is connected with.

Due to the strategy used by BLE Mesh, the messages must be exchanged in the scope of so-called models. Each model is responsible for managing the messages of a certain structure and semantics. A model defines a set of states, state transitions, state bindings, messages, and other associated behavior. Each node must support one or more models, and these model or models define the node's functionality. There are two basic types of BLE Mesh models. Special Interest Group proposes SIG models [21] as a set of well-defined models of fixed functionality (starting from such simple models as Generic ON/OFF, to quite more complicated such as battery-level monitoring). These models are usually implemented as a part of the BLE Mesh library. The second group of models is called vendor models. Vendor models use a slightly different mode of identification of messages (longer identifiers). Thus they are sometimes less efficient. However, these models are not standardized, so in theory, they may represent an arbitrary set of additional services (functions) of the node, prepared by the network designer.

In the next section, we describe our reasonable for proposing the implementation of a certain set of BLE Mesh models, in turn, to implement REST messaging in the mesh network.

V. Implementation issues

As described above, one has to propose a model to exchange the information in the scope of the BLE Mesh network. The model comprises both message syntax (i.e., data length) and semantics (i.e., data type and location in the buffer). We decided to provide two implementations: a new SIG model and a vendor model. The implementation was based on Nordic Semiconductor processors (namely nrf52840) and NS BLE Mesh library, and Espressif processors (ESP-32 WROOM-E, ESP-32 S3) programmed within the IDF framework. In addition, we adapted the Nordic library for BLE Mesh for Android smartphones to administrate/test the network. The adaptation aimed in implementing the same models we designed for BLE Mesh as an extension to Nordic library. Please note that the main reason for choosing ESP-32 and nrf52840 processors was based on the fact they have a built-in BLE unit. Obviously, the solution would work for non-BLE processors such as most of the STM family with external BLE units.

It was soon detected that the choice of implementing the model as a new member of the SIG-model group was not justified. Even if, in theory, such implementation is more efficient due to shorter identifiers, it was soon detected that the library lacks documentation about some programming tricks applied by SIG programmers. For example, fixed tables of semaphores and fixed maps of model identifiers were used and indexed by model identifiers. As a result, the library itself needed to be rewritten and re-compiled. Such an adaptation should be performed every few months as a new library version comes unless SIG agrees to incorporate the model into the set of their models (which has not been planned so far).

Thus, the work concentrated on implementing a vendor model, which was less efficient, however, with no reason to update basic libraries. In contrast to any of the SIG models, the model is based on strings of characters of variable length (textual messaging). Thus, the model is quite generic, which is not the case with the existing models, and the semantic analysis of the message is to be performed at the application level. To this goal, a dedicated interpreter has been designed. Fetching a textual REST request at the input, the interpreter produces a union at the output containing the type identification (a node of the graph of the hierarchy of all types), subtype, and a list of query parameters in the form <name, value>, where “name” is a string of characters, and “value” is a number or a string of characters. “Subtype” needs more explanations. This parameter is a difference between the most-specialized type name within the hierarchy of types and the type identifier used to formulate the REST request. For example, the identifier “kettle/silver” identifies “kettle” as the basic type (as this is the most specialized node in the hierarchy of types addressing “kettle” identification), and “silver” as a subtype. Using subtypes is similar to the usage of individual names of real-world artifacts connected to the mesh nodes. It is up to each node to interpret not only the type name but also the subtype, for example, to distinguish the artifacts of the same type but different meanings or locations (such as “tv/kitchen” and “tv/mainroom”). Please note that type and subtype names conform to the REST approach to resource naming.

Queries are encoded according to the REST standard (so-called URL-encoding schema). The query addresses the detailed functionality of a node and is also interpreted at-the-place, similar to the traditional REST approach. The query elements (names and domains/formats) are not standardized. Instead, it is up to the caller to formulate the query so the mesh nodes can understand it.

Note that our textual BLE Mesh model somehow replaces any previous models. Formulating the types and queries is relatively straightforward in such a way that they are equivalent to any other call. For example, instead of the time-synchronization model, one may apply “any?setTime=12:34:56” message, asking for setting the time at each node (“any” is the root of the hierarchy of all types, cf. Fig. 2).

We also found that sometimes knowledge is needed about the existence of the nodes of a certain type in the network. Returning to one of the previous examples, a “ventilator” should inform all the "ovens" that it is possible to force the airflow, in case it becomes too hot, or a smell is irritating. To this goal, it is enough to periodically broadcast a message stating each node's type and current state. For example, the ventilator could broadcast "any?airFlow=1" informing any other node about its current state. Again, the interpretation of such information depends on each node's needs and capabilities, and the semantics of the query elements needs to be known for all the nodes interested in such information. For the rest of the nodes, such messages will be ignored as pointing to an unknown action to be done, similar to the traditional interpretation of REST requests.

VI. Sample implementation – a kitchenware application

As already mentioned, our implementation targeted "an intelligent kitchen" idea. We took several assumptions to characterize the needs of such specific networking better:

- in general, within a single network, we use one device of a given type, such as a "kettle", a "microwave_oven" etc. It is possible to use subtypes to differentiate several instances (entities of the real world) of the same type, as described in the previous section, if needed. If not parameterized, all devices of the same type react to the incoming message (such as the message "lamp?on=1" will switch on all the lamps in the location);
- we use classical RESTful usage of the mesh network as a whole, via a WiFi gateway implemented in one selected mesh node (in our case – a kettle) and a dedicated Android application connected to the same WiFi network; the Android application is equipped with some extensions to facilitate user interfacing, such as voice analysis/synthesis;
- we apply mapped REST calls (broadcasts) inside the network to control the devices in a predefined manner (any node knew its type and meaning of the query elements);
- there is no centralized directory of network services; however, some devices (especially a smartphone) may specialize in grouping/providing some information about some other devices, if needed;
- we rely on the security and relaying mechanisms from BLE Mesh (in particular, the provisioning of each new node);
- we apply periodic broadcasts of each device's state (message content depending on the device's functionality and type) to enable Android application planning and controlling the kitchen activities as a whole.

As already mentioned, we use a smartphone (Android-based) as a provisioning center and several network nodes based on processors with BLE support, mainly Espressif ESP 32 WROOM-32D/E, ESP-32 S3 (BLE 4.x), and Nordic Semiconductor nrf52840 (BLE 5.x). Selected Espressif nodes also serve as WiFi gateways. These nodes broadcast their IP addresses to all the other nodes. Note that the overall WiFi security is not broken here, as the broadcast messaging is encoded with BLE Mesh keys, and thus it is readable only by the members of the BLE network.

During the tests, we found that the smartphone application based on Nordic Semiconductor's library is insufficient for contacting the device. The library itself is huge, and the application consumes a lot of energy for transmission (both BLE and WiFi connections are active all the time). Thus we apply a mixed mode – the application listens to BLE broadcasts and sends all the requests to the network to any of the available WiFi gateways. Even if strange at the very first view, such a mixed mode was found as quite efficient and easy to implement. The communication mode depends on end-user strategy: WiFi calls are less efficient but more straightforward to implement (standard HTTP calls and REST messaging), BLE messaging needs a separate implementation of the new mesh model, but then the IP address is not necessarily known. The choice should be left to the users.

The above-mentioned application was also used as the main provisioning center for the network. As already noticed, we soon found such a way of provisioning as non-efficient, and we are working now on a new approach to provision the new nodes by any existing node from the network. Once completed, such distributed provisioning will eliminate the need for a smartphone as a network node. This work is not finished yet; it also needs some hardware extensions, such as WPS buttons known from WiFi access points. Once the work is finished, it is to be described in a separate paper. To our best knowledge, no proposal exists for dynamic provisioning within a single mesh network, where any node may act as a provisioner. Please note that we cannot rely on a single predefined provisioner, as this node may be temporarily out-of-network or even gone, thus preventing any new node to join. The above problem is also linked to the so-called "newcomer" problem, i.e., how to find an entry point to access the network for the first time. This problem also needs particular attention; we plan to work on it using intelligent BLE beacons.

Our implementation aimed to design an "intelligent" kitchen. Thus, besides the smartphone as a provisioning center, we linked the mesh nodes with the following kitchenware: a kettle (this device was equipped with a touch screen to serve as a main network node), an induction plate, a ventilator, a lamp, radio with an MP3 player.

The touch screen of the kettle could be converted to act as an interface of any network node to (1) facilitate the user interface (the interface was unified for all the devices regardless of their type, but taking into account their specificity), and (2) limit the costs of the "intelligence" of the devices to a reasonable minimum. We implemented the following automatic messaging:

- the radio automatically increases its volume while the kettle is finishing boiling the water,
- the ventilation is started after 10 minutes of using an induction field with a level greater than 50% of the maximum,

- the kettle sends an alert to the MP3 player when hot water is ready,
- the ventilation is stopped if none of the devices was used (i.e., activated) for the past 30 minutes.

The above functionality may be programmed by the producer of the devices, not necessarily the end-user. This is only a sample set of actions to be performed automatically by the network. Any new device may be incorporated at any time with no changes in the code/variables of any other device. There is no centralized directory of services, global coordinator, "main" node (maybe except the kettle – but only for economic reasons), etc. There is also no need for the "installation" of any new device (except for the provisioning process we plan to improve, as already mentioned).

Please note that our network is neither a classical sensor network nor a MANET one. This is also not quite an ad-hoc network. Even if the nodes may be switched on/off at the temporary base and coincidentally, apart from such switching, they are relatively stable – always at the same place, and usually with the same (fixed) functionality. Some nodes are used frequently, some all the time, some at request, and some rare, but the set is not evolving very often (unless a new device is bought or an old device is broken). Thus, our approach is well-suited to the above circumstances.

VII. Comparison with previous work

While we started the research for similar work to be compared with our proposal, we found that most of the existing proposals concentrated on the "smart home" idea, with an engineer's design of ready-to-market products. Some of these proposals were related to patent applications, demonstrating a growing need to provide such solutions on the market. Although very popular in the scope of client-server architecture and classical applications, we discovered that the REST idea was hardly used for controlling the behavior of home appliances and systems except the proposals based on WiFi traffic (classical REST applications). To our best knowledge, no single proposal exists for using REST like addressing in Bluetooth-based mesh networking. The existing proposals aimed in using specialized, centralized directories of services capable of mapping WiFi calls to Bluetooth direct (paired) communication. Thus, we compared our proposal with similar proposals for efficiently managing "smart" systems and devices used at home, especially in the kitchen.

Known solutions for communication and control of home devices use a centralized home network in which the entire network transmission is supervised by a center, which is most often a specialized router, sometimes a smartphone, or the most advanced, always-on home device, for example, a refrigerator. In a network with such a topology, all data sent over the network must be sent by the central node, and each device must be installed and registered in this node before its first use.

Chinese patent specification CN109218098A [22] discloses a smart home control method, a radio transmission network gateway, and a smart home control system. The gateway is the primary authorization center and the primary point that limits the functionality and capacity of the entire network. It is a typical centralized solution that requires installation in the network and registration of each new device and uses the traditional method of addressing the devices.

Another patent specification CN109088994A [23] discloses a solution in which a smartphone is used to establish a connection with one device equipped with a Bluetooth communication module at a time. The application can transcribe the command given via the graphical or voice interface to the commands sent via Bluetooth to the currently connected home device. The application also acts as a central directory of available devices that must meet very strict requirements as to the type and method of data transmission, which in practice limits the number of such devices only to the list prepared for the purposes of this invention. A severe functional limitation of this system is also that home devices cannot directly exchange any information with each other. The data must be transferred only to and from the smartphone.

The invention described in document CN109981776A [24] concerns a system that solves the problem of the limitations of the classic Bluetooth communication channel, i.e., "exclusive" operation over an established link in this type of transmission. Each device has several Bluetooth transmission modules, the first of which is used to synchronize access to the others. In this way, by establishing connections in the other modules for a while, one can transmit data between any pair of network devices through them. The biggest disadvantage of this solution is the necessity to install many communication modules in each device. A similar solution is depicted in [25].

Document CN111585855A [26] discloses a system that uses a smart wireless router integrated with a WiFi module, an infrared module, a ZigBee module, and a Bluetooth module, which enables address communication with any device available through supported forms of communication. The system does not provide for direct communication between these devices without the use of the above-mentioned router.

Contrary to all the above-presented (and similar) proposals, this paper describes a system and method of communication with intelligent home devices via a network without a central point, in which each device is equally privileged. Automatic communication among the devices is possible without human intervention. In addition, external requests are formatted in a way to replace node addresses with semantic names of functions performed by devices.

The purpose has been achieved with the use of a Bluetooth mesh topology network, in which communication among network components is possible without the need to involve the central unit. In such a network, each network device is equally privileged and can communicate with any other device directly or via any other network component. Selected devices, equipped with alternative communication modules of a different type, in particular, connected to the home WiFi network, can be used as input/output gateways for communication outside the mesh network. Transmission in the network occurs in a broadcast mode, in which one node sends information that is then received by all other nodes.

No device is individually addressed in the proposed solution, so no central directory is needed. Installing each new or replaced device or monitoring the device status is unnecessary. The transmitted signals contain digital information formatted in accordance with the REST software architecture style with respect to the fact that the addresses of devices are to be replaced with semantic names of functions performed by devices. For each device, one can define any set of functions for which that device will be responsible. The naming of the functions corresponds to the REST resources naming.

The proposal responds to the idea “different locations, similar usage”. There is no installation needed, such as when users buy new equipment, these devices are, from the very start, ready to use. Moreover, we detected un unexpected add-on while working with the smart kitchen: a possibility of targeted, personal marketing for non-existing devices. Once a requested functionality is not achieved, any other device (in our case it was the most complicated and advanced one – a kettle) may detect this fact, contact the cloud for a possible solution and broadcast some advices in the local network. As a consequence, the users are informed in JIP (just-in-place) and JIT (just-in-time) manner, thus increasing the probability of taking a decision and buying the missing equipment.

Our semi-persistent services are a response to fixed needs of the users changing places (such as several locations of a single family, or family helper), or changing organization of a mesh at a single place (such as a home or a kitchen) without a need for registration and tracking the current status of the services. Moreover, the same application may be used to control a smart kitchen, and near-by – a home audio-video system.

VIII. Conclusions

In the paper, we proposed an adaptation of the popular REST approach to BLE Mesh networking and address-free traffic. The idea enables a system and method of communication with home appliances and components of home infrastructure, such as household appliances, audio/video, lighting, heating, and air-conditioning equipment, in order to control these devices in a decentralized manner.

We use double-mode communication with home devices connected via microcontrollers to the BLE Mesh network, and WiFi. Connecting a smartphone or a computer device (e.g., a laptop or a tablet) is also possible. Gateway nodes automatically map REST messaging from WiFi networking to internal BLE messaging, conforming to REST strategy and using a hierarchy of types instead of resource names. The types enable semantic interpretation of the REST messages to be used dynamically for the such variable environment as a mesh network.

Messaging of REST-compliant communication is implemented as a broadcast transmission in the scope of a dedicated vendor model of a BLE Mesh network, implemented for the popular BLE microcontrollers as well as Android smartphones. The model is equipped with an interpreter of the incoming messages based on the semantics of the type (basic part of REST resource name), subtype (additional elements of REST resource name), and REST query complemented with some network-specific parameters. The query is provided as a list of parameters of the "name-value" type, where "name" means a command to activate a function, and "value" is a parameter of such a command. The requested functions specified in the information sent are activated for the real kitchen device associated with a given resource type, and parameterized by the query. Each device is responsible for (1) filtering all the incoming messages by their types and the accordance with the type(s) declared for the device, and (2) interpreting the query parameters. The filtering and the interpretation are programmed in the control code of the devices, and preferably switched on/off by end-users.

The devices associated with the network nodes do not have any information about the other devices on the network, including whether the device is active on the network or not. A message with a command sent by a device to an inactive device on the network does not affect the operation of the device sending the command. In the embodiment of the invention described above, the induction cooktop will not stop cooking if the cooker hood does not turn on. Still, if the cooker hood is active on the network, it will start automatically if needed.

The devices on the network are identified only by the name of the currently assigned resource and the set of functions assigned to that resource. The address or location of the device is not required for communication between devices on the network. The device cooperation can be programmed by placing appropriate REST messages in the device microcontroller code. Replacement of one device model with another will not require reprogramming these microcontrollers, and the network as a whole will work the same despite the changes.

The system is fully implemented and tested for the devices produced by Polish biggest kitchenware manufacturer Amica. It was also a base for a European patent application [27].

As for future work, we plan to extend the proposal to address the problem of the “newcomer”. If the users visit an ad-hoc location for the first time, they are not informed about the possible services to be accessed there. Thus, some experiments are needed to determine which services are available. To minimize the time spent for these experiments, we plan to include a so-called "advertisement channel" and

BLE beaconing [28] to broadcast some information for early detection of the services and their types.

We also plan to apply one of the well-known ontologies of IoT devices to provide some generic services in public places. An obvious candidate for such a service is a thermometer, but also UV-meter and PM-* detectors, to be used primarily to protect people with asthma and similar diseases. So far we concentrated on smart kitchen, but the number of ontologies (as well as the level of their complexity) for this application area is limited. If, however, we plan to extend our approach to some public places and common devices, using such external ontology is a must. Selecting given ontology depends on the application area, but our approach makes it possible to address as many different ontologies as it is needed, by a selection of an "optimal" type hierarchy.

We also plan to optimize the process of validating access to the BLE Mesh network, so-called provisioning [29], aimed at exchanging encryption keys. So far, a dedicated mesh node called a provisioner has been used for this goal. However, this node may be temporarily inaccessible for many reasons, thus preventing new users from entering the network. Thus, we plan to apply the provisioning function to any node and dissipate the necessary information among the other nodes, dynamically voting for the best "candidate" for at-the-moment provisioning.

REFERENCES

[1] R. Fielding, Representational State Transfer (REST), Ph.D. dissertation, [Online] available: https://www.ics.uci.edu/ ~fielding/pubs/dissertation/ rest_arch_style.htm, 2000

[2] L. Gupta, What is REST - REST API Tutorial, [Online] available: https://restfulapi.net/, 2022

[3] What is a URL?, Mozilla documentation, [Online] available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/ Web_mechanics/What_is_a_URL, 2023

[4] R. Ramanathan, J. Redi, Overview of ad-hoc networks: challenges and directions, IEEE Comm, Volume 40, Issue 5, DOI 10.1109/MCOM.2002.1006968, 2002

[5] S. Al-Sultan,M. M. Al-Doori, A. H. Al-Bayatti, H. Zedan, A comprehensive survey on vehicular Ad Hoc network, Journal of Network and Computer Applications, Volume 37, Pages 380-392, 2014

[6] M. S. BenSaleh, R. Saida, Y. Hadj Kacem, M. Abid, "Wireless Sensor Network Design Methodologies: A Survey", Journal of Sensors, vol. 2020, Article ID 9592836, [Online] available: https://doi.org/10.1155/2020/9592836, 2020

[7] A. O. Bang, P. L. Ramteke, MANET: History, Challenges, and Applications, International Journal of Application or Innovation in Engineering & Management, Volume 2, Issue 9, ISSN 2319 – 4847, 2013

[8] N. Temene, C. Sergiou, V. Vassiliou, C. Georgiou, A Survey on Mobility in Wireless Sensor Networks, Ad Hoc Networks, Volume 125, 1(February), [Online] available: https://doi.org/10.1016/ j.adhoc.2021.102726, 2022

[9] R. Zagrouba, A. Kardi, Comparative Study of Energy Efficient Routing Techniques in Wireless Sensor Networks, Special Issue Wireless IoT Network Protocols, Information, 12(1), 42; [Online] available: https://doi.org/10.3390/info12010042, 2021

[10] M. Wang, Y. Chen, X. Sun, F. Xiao and X. Xu, "Node Energy Consumption Balanced Multi-Hop Transmission for Underwater Acoustic Sensor Networks Based on Clustering Algorithm," in IEEE Access, vol. 8, pp. 191231-191241, doi: 10.1109/ACCESS.2020.3032019, 2020

[11] Z. Lv, D. Chen, H. Feng, W. Wei, H. Lv, Artificial Intelligence in Underwater Digital Twins Sensor Networks, ACM Transactions on Sensor Networks, Volume 18, Issue 3, Article No. 39, pp 1–27, https://doi.org/10.1145/3519301, 2022

[12] A.M. Rahmani, P. Liljeberg, J.-S. Preden, A. Jantsch, Fog Computing in the Internet of Things - Intelligence at the Edge, Springer International Publishing AG, [Online] available: https://doi.org/10.1007/978-3-319-57639-8, 2018

[13] J. Elson, D. Estrin, An Address Free Architecture for Dynamic Sensor Networks, Research Gate, [Online] available: https://www.researchgate.n et/publication/2618136_An_Address-Free_Architecture_for_Dynamic_Sensor_Networks/citation/downlsoad , 2000

[14] A. Cilfone, L. Davoli, L. Belli, G. Ferrari, "Wireless Mesh Networking: An IoT-Oriented Perspective Survey on Relevant Technologies". Future Internet. 11 (4): 99, doi:10.3390/fi11040099, 2019

[15] Y. Junjie, Y. Zheng, C. Hao , L. Tongtong , Z. Zimu , W. Chenshu, A Survey on Bluetooth 5.0 and Mesh: New Milestones of IoT, ACM Transactions on Sensor Networks, Volume 15, Issue 3, Article No. 28, pp. 1–29, https://doi.org/10.1145/3317687, 2019

[16] M. Baert, J. Rossey, A. Shahid, J. Hoebeke. The Bluetooth mesh standard: An overview and experimental evaluation. Sensors (Basel, Switzerland) 18, 8(July), p. 2409, 2018

[17] M. Collotta, G. Pau, T. Talty, and O. K. Tonguz. 2018. Bluetooth 5: A concrete step forward toward the IoT. IEEE Communications Magazine 56, 7 (July), pp. 125-131, 2018

[18] M.R. Ghori, T.-C. Wan, G.C. Sodhy, Bluetooth Low Energy Mesh Networks: Survey of Communication and Security Protocols, Sensors, 20, 3590, [Online] available: https://doi.org/10.3390/s20123590, 2020

[19] A. Lacava, V. Zottola, A. Bonaldo, F. Cuomo, S. Basagni, Securing Bluetooth Low Energy networking: An overview of security procedures and threats, Computer Networks, Volume 211, 5, 2022

[20] M. Noura, M. Atiquzzaman & M. Gaedke, Interoperability in Internet of Things: Taxonomies and Open Challenges, Mobile Netw. Appl. 24, 796–809, [Online] available: https://doi.org/10.1007/s11036-018-1089-9, 2019

[21] Bluetooth Mesh Models - A Technical Overview, official Bluetooth documentation, [Online] available: https://www.bluetooth.com/ bluetooth-resources/bluetooth-mesh-models/, 2023

[22] Shenzhen Zhizhen Science And Technology Co Ltd, A kind of connection and configuration method of home gateway, China patent application CN109218098A, [Online] available: https://patents.google.com/ patent/CN109218098A/en?oq=CN109218098A, 2019

[23] Lanzhou University of Technology, Based on smart phone and single-chip microcontroller intelligent miniature household method, China patent application CN109088994A, [Online] available: https://patents.google.com/patent/CN109088994A/en?oq=CN10908899 4A+, 2018

[24] Foshan Shunde Midea Washing Appliances Manufacturing Co Ltd, Intelligent control equipment, the networking control method of household appliance and system, China patent application CN109981776A, [Online] available: https://patents.google.com/patent/ CN109981776A/en?oq=CN109981776A+, 2019

[25] K. Takada et al., Communication apparatus, communication system, notification method, and program product, United States Patent US 9,497,629 B2, [Online] available: https://patents.google.com/patent/ US9497629, 2014

[26] Bowei Technology Co ltd, Intelligent wireless router and intelligent home system, China patent application CN111585855A, [Online] available: https://patents.google.com/patent/CN111585855A/en?oq= CN111585855A+, 2020

[27] J. Rykowski, T. Jenek, W. Switala, System and method of communication with home devices, European patent application EP 4 132 034 A1, [Online] available: https://data.epo.org/publication-server/rest/v1.0/publication-dates/20230208/patents/ EP4132034NWA1/ document.pdf, 2022

[28] Estimote Inc., How do beacons work?, Estimote dcoumentation, [Online] available: https://community.estimote.com/hc/en-us/articles/360002656512-How-do-beacons-work

[29] K. Ren, Provisioning a Bluetooth Mesh Network Part 1, Bluetooth documentation, [Online] available: https://www.bluetooth.com/blog/ provisioning-a-bluetooth-mesh-network-part-1/