Chapter

Optimizing a Centralized Control Topology of an IoT Network Based on Hilbert Space

Jesús Jaime Moreno Escobar, Oswaldo Morales Matamoros,
Hugo Quintana Espinosa, Ricardo Tejeida Padilla
and Ana Gabriela Ramŕez Gutiérrez

Abstract

An Internet of Things network (IoTN) is composed of many small devices or nodes located in homes and/or offices, to be operated through the Internet from anywhere, making these devices smarter and more efficient. For improving the efficiency of an IoTN, in this chapter an optimized fractal algorithm (OFA) was proposed for designing a centralized control topology of an IoTN, whose nodes are distributed according to the Hilbert space-filling fractal. We developed the OFA to find the best nodes where a smart home device can find the highly reliable link between its neighbors by a software-defined network (SDN) with a target coverage since OFA only considers reliable links among devices. Finally, through laboratory tests and computer simulations, we demonstrated the effectiveness of this proposal by using a large amount of IoT devices making them more efficient operating systems. The quality of service (QoS) is a challenge that guaranteed the level of service delivery to an IoTN, so that OFA takes less time to reach its destination after it is generated by its source, increasing the probability that the target node can recover the original packet before the lifetime expires.

Keywords: smart home, IoTN, centralized control topology, optimization, Hilbert fractal algorithm, quality of service

1. Introduction

In the coming fourth industrial revolution, several technological fields are impacting all disciplines, economies, and industries, such as artificial intelligence (AI) and the Internet of Things (IoT). Artificial intelligence permits machines to learn and adapt to different situations and problems. Due to its great versatility, AI can work in conjunction with the Internet of Things, allowing smart home devices or microcontroller units (MCUs) to create many Internet of Things networks (IoTNs) to be able to collect, process, and share data of different nature that flows through the network. It is expected that by 2025 26–30 million MCUs in homes and offices will be connected as networks to the Internet and equipped with sensors, processors, and embedded software.

As a network, IoTN behaves as a collective entity which does not depend on a central control or hub. Therefore, it is necessary to design IoTN topologies that are not subject to the performance of a single MCU or node for guarantying the quality of service (QoS) requirements due to an efficient connectivity based on a bandwidth that links each of MCUs with their immediate neighbors at 300 Mbps, in order to ensure that all of them really share their parameters when they are operating. This implies to design networks with shared parameters between their nodes and where the locality is preserved and limited the maximum average distance between the MCUs by using alternative tools, such as space-filling curves. The Hilbert fractal is a continuous space-filling curve whose locality-preserving behavior is better than that of Z-order curves, because the distance between each node in a Hilbert curve does not fluctuate, whereas that distance in a Z-order curve does fluctuate.

Accordingly, in this chapter an effective and a reliable optimized fractal algorithm (OFA) for extending the range of transmission of a given IoTN in an intelligent, adaptive, and dynamic way is proposed. The OFA finds a path from a source to a target to optimize the links by considering the quality of service constraints such as end-to-end reliability and delay. In this way, with OFA we pretend to achieve the best two MCUs where each smart home of the IoTN can transfer or share its parameters. Finally, through laboratory tests and computer simulations, we demonstrate the effectiveness of our approach by means of a fractal routing in IoTN, using from 16 to 64 smart home devices for real-time monitoring of different parameters, in order to make more efficient the Smart Home Automation by reducing the IoTN energy consumption.

The further sections of this chapter are organized as follows. First, in Section 2 we explain the relation between smart home devices distributed in networks and the energy efficiency. In Section 3 an explanation of the theoretical definition of the Hilbert space-filling fractal is given. OFA scheme is defined in Section 4, while in Section 5, the simulation and the comparison of the performances are announced, in order to verify the correction and feasibility of the proposal. Finally, the arguments and discussion of this chapter are analyzed in Section 6.

2. Merging smart home and software-defined networks

A house is smart when it has installed a series of electronic systems, sensors, and devices, so that any person can easily control it, even from a distance, and the house performs certain actions on their own.

An example of this would be the management of heating and air conditioning, so that our home always has a comfortable temperature, whatever the time of year.

This can be achieved by installing temperature sensors and connecting them to a computer or a dedicated device, which once we have programmed it, deciding which temperature is the best for the residents, sends orders to the heating or air-conditioning control devices. This automation is what is called home automation.

In this way, home automation has been around for a long time, as we can link various automations through domestic wiring, sometimes with cable inside the walls. The technological advance has given us wireless connectivity and miniaturization, which have allowed the extension of the concept of home automation to many other aspects, even outside the home.

Currently, and under the definition of the smart home, there are devices that allow to control almost everything imaginable, either automatically or facilitating human control.

In this sense, we can say that the smart home is the result of the intersection between architecture, interior design, and the most advanced technology, which allows the automation in an unattended way of certain domestic tasks, but also with the condition that there can be supervision or fixed objectives on the part of the residents.

Thus, a smart home seeks to:

- Increase our security cameras or motion sensors which are elements that warn us of intrusions and suspicious movements even when we are absent.

- Provide us more time. By automating certain tasks, we stop thinking about them. If we wake up every morning at a certain time, the same home (through its artificial intelligence) can raise the blinds and start to heat the coffee, so we can save this time to get to work a little earlier or simply to sleep 5 more minutes.

- Allow us remote control. Turning on the air-conditioning unit 10 min before we get home so that when we enter it.

- The energy saving with multiple sensors scattered around the home are more reliable than the human sensation, where exactly the energy needed to heat or cool the house is used.

The latter statement guides us to consider as one of the most important issues the way to save energy among sensors, namely, how efficient is the energy consumption. In this way we identify energy efficiency as the most critical challenges in IoTNs because the nodes or smart home devices in such networks have limited resources. In this way, software-defined networks (SDNs) are a way to approach the creation of networks in which control is detached from the hardware and given to a software application called a controller. So, SDNs, also known as programmable and automated networks, are presented as a proposal that provides greater speed, agile infrastructure, and better costs in cloud IT platforms; it is urgent to respond to the dynamism of the applications required by the user.

When a packet arrives at a switch in a conventional home network, the rules built into the proprietary firmware of the switch tell the switch where to transfer the packet. The switch or router sends each packet to the same destination on the same path and treats all packets in the exact same way. In this proposal, intelligent routers designed with specific MCUs are sophisticated enough to recognize different types of packages and treat them differently, but these router can be only the seed or the begging of the home network. Furthermore, in an SDN, a home network administrator can shape traffic from a centralized control console without having to touch individual smart devices, in this case sensors. The administrator can modify any rule of network switches when necessary—giving or removing priority or even blocking specific types of packets with a very detailed level of control.

This is especially useful in a multi-tenant architecture for cloud computing because it allows the administrator to handle traffic loads flexibly and more efficiently. Essentially, this allows the administrator to use fewer and more *intelligent routers* and have more control than ever over the flow of all the smart devices of the network traffic. Nowadays, the most popular specification for creating a software-defined network is an open standard called OpenFlow. OpenFlow allows network administrators to control routing tables remotely.

In addition, home networks have not changed about 20 years ago; unlike programs, we are facing a paradigm shift to which we must adapt. The development of

SDNs began in 1990, where programmable functions are included in the network; in 2001–2007, the control and data planes were improved, improving with this innovation. From 2007 to 2010, it was implemented. The OpenFlow API, is presented as an open interface, presents various ways the separation of the control plane and data to be scalable, practice where virtualization played an important role in this SDN evolution.

The progress and evolution of SDNs underlies because traditional home routers cannot respond to unpredictable traffic patterns much less to peak demand. In addition, there would be two alternatives to climb to a more expensive home network by spending more time in configuration or adapting to a dynamic home network. SDNs are suitable for people that require rapid changes in the short term, which happens in a conventional home where mobile devices are added and disappear arbitrarily over time. Currently a conventional home user uses social networks in many devices, which are in high demand or require sudden changes, for example, not only working on geographic traffic but environmental variables such as temperature or humidity, development of mobile devices, virtualization of network devices and sensors, and cloud services.

In this way it is important to point out that SDN is an architecture approach, not a specific product, which has traditionally been thought of as the virtualization of data center networks.

This means separating the management of the control plane from the home network devices from the underlying data plane that forwards the home network traffic. The use of a software-defined system to control this disaggregation provides many benefits, including greater network management flexibility and the ability to more easily implement high-precision security policies.

However, Kindness points out that smart home network operators think of SDN with a too narrow focus, although there has been an evolution in the SDN market in recent years, driven by the increase in demands on the network.

To meet these new challenges, the underlying technology that drives SDN has also been applied to other network areas such as the homes with a lot of sensor or smart devices.

SDNs emerged in early 2010 out of mere necessity. Many networks today were designed for client–server applications that run on a non-virtualized infrastructure. The purchase of Nicira by VMware in 2013 was considered a turning point for the SDN industry, as it turned the virtualization giant into a network provider. Today, VMware's NSX SDN product is based on that technology. Cisco Application Centric Infrastructure is the basis of its SDN offer. Many other companies, such as Juniper and Arista, also offer their solutions. IDC estimates that the SDN market has grown from an industry of 406 million dollars in 2013 to over 6600 million dollars in 2017. The consulting firm anticipates that the SDN market will continue to do so at a compound annual growth rate of 25.4%, reaching about 13,800 million dollars in 2021.

A survey conducted in 2017 by the Network World publication among 294 network professionals found that 49% of them are considering or actively piloting an SDN implementation; 18% already have an SDN installed.

Furthermore, IDC has identified a handful of major use cases for SDN today:

• To maximize investments in server virtualization and private cloud

• To enable network programming

Although so far many SDN deployments have focused on the data center network, Kindness, the Forrester analyst, states that the future of SDN will be defined by how this technology is used outside of the data center.

There are a variety of factors that will continue to pressure network operators, including the increasing use of public cloud computing, the attack of network traffic created by the Internet of Things, the proliferation of a mobile workforce, and an increasing number of distributed branches. Given this, SDN will play a role in configuring the next generation of networks for each of these use cases.

And there is already evidence of this in actual use: Software-Defined Wide Area Network (SD-WAN) is a software management platform to control access to the remote offices or branches of an organization.

In the past, home customers had only one connection to their branches; today SD-WAN allows companies to add several types of network connections to a branch and have a software management platform that allows high availability and automatically prioritize traffic.

SD-WANs can save a home customer's expense by installing expensive custom WAN acceleration hardware allowing them to run a software overlay on less expensive hardware.

Expects highlight that the SD-WAN will become a market of 6 billion by the year 2020.

In addition, software-defined IoTNs (SD-IoTNs) can be technically sorted in nine classifications:

1. Security: identifying the malicious user and their activities, namely, global overview of a device's status in the home network.

2. Routing: the network data and information is efficiently transferred.

3. Mobility: due to external forces, any node of the home network can be physically moved.

4. Reliability: the ability to be, almost always, in an operational state under adverse circumstances, usually after failures. Operating state is understood to be that in which the SDNs are capable of performing a specific operation.

5. Management: maintenance, network configuration, and provisioning.

6. Quality of service: it is a set of service requirements that SDN must accomplish when routing a data flow. It can be implemented in different situations, to manage congestion or to avoid it. It allows to control some significant characteristics of packet transmission. These characteristics can be specified in quantitative or statistical terms such as bandwidth, latency, jitter, and packet loss in the network, ensuring a pre-established degree of reliability that meets the traffic requirements, depending on the profile and bandwidth for a given data flow.

7. Wireless power transfer: energy sensor node can be transmitted to other nodes through an appropriate transmitter.

8. Localization: many applications of IoTNs need the information of each node.

9. Energy efficiency: sleep scheduling approaches are designed for switching the nodes into idle state if their functionality is not required.

    In addition, this classification can be rewritten as follows:

    • Coverage control: control activates or deactivates the sensor nodes to cover a network region.

- Clustering: nodes into clusters and a head node for each.

- Lifetime: possibility to utilize the node capabilities for a longer period of time.

Accordingly to SD-IoTN paradigm, no matter the sort of topology, any sensors inside a smart home network are connected individually to a central router, and they were almost never interconnected among them. Thus, the devices furthest from the router spend more energy than those that are closest to it. In addition, any communication between sensors must necessarily go through a centralized router, even if they are a foot away from each other.

In this sense, the main proposal of this chapter is to communicate with the nearest sensor using a non-centralized, rather a distributed and dynamic, topology that is reorganized according to a fractal function based on the Hilbert scanning.

3. Hilbert space-filling fractal

Mathematician B. Mandelbrot coined the fractal term in his pioneer work *The Fractal Geometry of Nature* [1], describing a fractal as an irregular or fragmented geometric structure that can be divided into parts: each of which is (or approximately) a smaller-size copy of the whole. He also pointed out that many fractals are found in the nature forming irregularly shaped objects or spatially nonstandardized phenomena in nature that cannot be attributed to Euclidean geometry, such as mountains or blood vessels, with fractional or non-integer dimension. From a mathematical point of view, fractals are a kind of composite geometric shapes which regularly display the property of self-similarity, such that a small segment of it can be reduced as a fractional scale replica of the whole [1].

All obtainable fractal objects in nature are generated from non-determined or random steps. Fractals generated by an iterative procedure, produced by consecutive dilations and conversions of a primary set, are deterministic. There are two properties attributed to fractals: self-similarity and space-filling. Self-similarity stands for a piece of the fractal geometry which seems to be like that of the total structure for all time, while the space-filling property means that a fractal outline can be packed in a limited region as the iteration increases without increasing the whole area.

Concerning the space-filling property, there are fractal curves that fill the plane that contains them in a specific order by continually changing direction or passing through each point that is in the defined space [2], such as the Hilbert fractal curve. The Hilbert space-filling curves are in a single layer that do not intersect; each point of them is at a constant distance unique to any other point, and these curves contain only one starting point and one stopping point in a single layer.

In 1878, mathematician G. Cantor demonstrated that there was a one-to-one correspondence between the unit interval $[0,1]$ and the unit square (plane) curve. In 1879, Netto showed that any such mapping could not be continuous. In 1887, Jordan defined a (plane) curve as the set of points $(\phi(t), \psi(t))$ where ϕ and ψ are continuous functions on a closed interval $[0,1]$, t is the time, and the curve is the path of a particle starting at $t = 0$ and ending at $t = 1$. In 1890, Peano discovered a space-filling curve. Thus the Peano curve must have multiple points, that is, points which are the images of two or more distinct values of t [3].

In 1891 Hilbert discovered another space-filling curve. Whereas Peano's curve was defined purely analytically, Hilbert's approach was geometric [2].

To construct Hilbert-type space-filling curves, lets denote the unit interval $[0,1]$ as $I = \{t|0 \le t \le 1\}$ and the unit square as $Q = \{(x; y)|0 \le x \le 1; 0 \le y \le 1\}$. For each

positive integer n, the interval I is partitioned into 4^n subintervals of length 4^{-n} and the square Q into 4^n subsquares of side 2^{-n}. A one-to-one correspondence between the subintervals of I and the subsquares of Q subject to two conditions is constructed: (i) adjacent subintervals correspond to adjacent subsquares with an edge in common, *adjacency condition*, and (ii) if at the $n - th$ partition, the subinterval I_{nk} corresponds to a subsquare Q_{nk}, and then at the $(n+1) - st$ partition, the four subintervals of I_{nk} must correspond to the four subsquares of Q_{nk}, *nesting condition*.

For each n the 4^n subintervals are labeled in their natural order from left to right. The correspondence between the intervals and the squares amounts to numbering the squares so that the adjacency and nesting conditions are satisfied. Hilbert's enumeration of the squares is shown in **Figure 1** for $n = 1, 2, 3$. The first square is always in the lower left corner, and the last square is always in the lower right corner. This means that the Hilbert space-filling curve starts at $(\phi(0), \psi(0))$ at $t = 0$ and ends at $(\phi(1), \psi(1))$ at $t = 1$. With the first and last squares of each partition determined, there is only one enumeration of the squares that satisfies the adjacency and nesting conditions.

The construction of a Hilbert space-filling curve is presented in **Figure 1**, in which the dotted square shows the area to be filled by the curve. This square is divided into four squares.

On the other hand, many space-filling curves can be produced as the limit of a sequence of polygonal curves, where the curves are generated via an iterative process, for example, a Lindenmayer system (L-system), in order to visualize such sequences of curves. A small part of the curve at one step of the iteration is close to a corresponding part of the curve at the previous step, and so it is natural to add frames that continuously interpolate between the curves of the iteration.

The Hilbert fractal can be generated by using specific rewriting rules of the Lindenmayer systems [5].
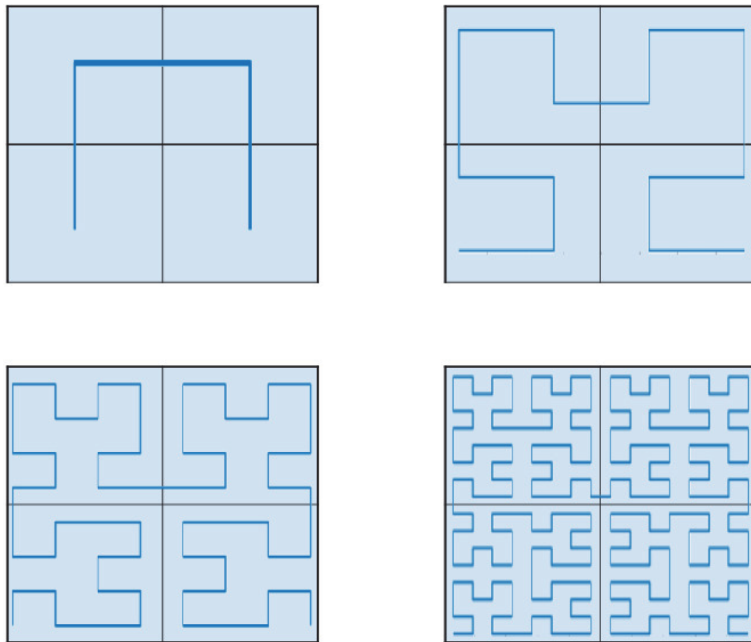

Figure 1.
First four stages in the generation of Hilbert's space-filling curve, axiom = R (see Section 4.3) employed by Hilbert in [4].

The L-systems, developed by Lindenmayer in 1968, emerged as a way to discreetly describe the development and growth of multicellular organisms [6]. An L-system is a string rewriting system. We have an alphabet of symbols V; a string of symbols ω defining the start of the system, or *axiom*; and a set of production rules P determining how we produce a new string from an old string. The *L-system* grammar is the collection of these three parts, $G = (V, \omega, P)$. This generates a sequence of strings by repeatedly applying the rules P, starting with the axiom ω.

To construct the Hilbert space-filling curve, we proposed the following L-system grammar:

$$G_{Hilbert=}(\{F, -, +\}, R, \{R \rightarrow +RF - LFL - FR+\})$$

Then the output of the L-system is the following sequence of strings for the first three steps:

Step 1:	$-LF + RFR + FL-$
Step 2:	$- + RF - LFL - FR + F + -LF+$
	$RFR + FL - F - LF + RFR + FL$
	$- + F + RF - LFL - FR + -$
Step 3:	$- + -LF + RFR + FL - F - +RF - LFL - FR + F + RF - LFL-$
	$FR + -F - LF + RFR + FL - +F + - + \text{RF-LFL-FR} + F + -$
	$LF + RFR + FL - F - LF + RFR + FL - +F + RF - LFL - FR$
	$+ - F - +RF - LFL - FR + F + -LF + RFR + FL - F - LF+$
	$RFR + FL - +F + RF - LFL - FR + -F + -RL + RFR + FL-$
	$F - +FR - LFL - FR + F + RF - LFL - FR + -F - LF + RFR + FL - +-$

Given the L-system for the Hilbert space-filling curve, it is possible to view the produced strings as instructions for building these curves in a turtle graphic fashion [7]. That is, we start somewhere on the plane, with a specified forward-facing direction. We read the input string from left to right and interpret the symbols as actions to be performed. For the L-system for the Hilbert space-filling curve, we make the following interpretations:

1. F, move forward one unit

2. +, turn left (counterclockwise) 90°

3. −, turn right (clockwise) 90°

In this chapter it is proposed to use the symbols L and R to represent a kind of small deviation from a forward step (starts with a left turn) and to represent a small detour in the other direction (starts with a right turn), respectively.

4. Methodology

The proposed methodology in this chapter is defined by the following passes:

• Hub MCU

• $LRN_i(t)$ and $LSN_i(t)$

- Hilbert fractal scanning

- Initial topology

It is also important to consider the general parts of the environment of the proposal, so any configuration of IoTNs consists of a wireless access point (WAP) and a β number of IoTN nodes which can be increased or decreased in its amount. In this case β embedded systems or MCUs that composed the IoTN, which shares parameters among sensors inside a given room or floor plan; β ranges dynamically are $1 \geq \beta > 4^n$. In this way, we also define $IoTN_N(t) = \sum_{i \geq 1}^{\beta} MCU_i(t)$, where $t = \{t_0, t_1, t_2, ...t_j\}$, i.e., a specific period of time (t) when the IoTN is defined as a set of MCU_i with at least one member up to 4^n members. Whereas n is the Hilbert fractal level needed to go from the first to the last MCU_i and it is expressed by $n = \lceil \log_4(\beta) \rceil$, here i represents the index inside the IoTN (t) of a given MCU, namely, MCU_1 is the first MCU, MCU_2 the second one, and so on.

It is important to emphasize that the IoTN has β MCUs in the time frame (t_j), but it can have up to $4^n - 1$ devices. If $\beta \geq 4^n$, the value of n is recalculated, and a new topology is reconfigured; otherwise the algorithm just adds a new node in the network, with the purpose that each node of the same network is linked or indexed by the curves that make up the Hilbert fractal. Subsequently, the effectiveness of the methodology exchange parameter throughout the network is measured. Finally, the point-to-point algorithms of Wi-Fi (P2P) and SoftAP are configured.

4.1 Hub MCU

It is important to establish that $\beta(t_j)$ IoTN devices are randomly distributed along a certain floor plan which forms a matrix γ with $2^n \times 2^n$ dimension. Namely, our proposal estimates a certain topology according to the β devices in the room in a certain period of time t_j. Then, the main WAP or hub is placed in the center of the main room, although can be placed anywhere within the room, to have connections at any direction. That is, every connection to the WAP will be the mainstream star topology. In this first generation of MNIs, a consecutive number will be assigned according to the speed of its link with the WAP. Thus, certain MCU_i has the best link so this device is labeled as $i = 1$ or MCU_1. Any floor plan had some walls, which attenuate the signal; that is why certain devices with shorter linear distance have a slower maximum speed than others that are further away from the WAP.

4.2 $LRN_i(t)$ and $LSN_i(t)$

$MCU_{i=1}(t)$ is identified as the main node in the IoTN. In this way, all the embedded systems both $IoTNe(t)$ and $MCU_1(t)$, enable Wi-Fi Direct mode and a full table with the bandwidth of the nodes are next to them, in order to generate a list of reliable nodes in a certain period of time t $(LRN_i(t))$. Every MCU generates a particular $LRN_i(t)$; then the proposed IoTN can generate 4^n LNRs at the instant t. In addition, every $LRN_i(t)$ contains the bandwidth of all $MCU_i(t)$ with which it establishes connection. In order to belong to the wireless sensor network, every $MCU_i(t)$ must connect to at least one link to another $MCU_i(t)$. All LRNs are shared, and $\sum_{i=1}^{4^n} MCU_i(t)$ knows the way to any node in the network, namely, everyone knows the topology of the network. In this way it is important to estimate the $LRN_i(t)$, but not all of these nodes are significant to be considered as the best option to establish a link, in which manner we define $LSN_i(t)$ as the list of significant nodes

(LSN) which is a vector that contains the best bandwidths of a certain $MCU_i(t)$. Algorithm 1 shows the methodology to estimate the $LSN_i(t)$ that needs i^{th} MCU and its $LRN_i(t)$. At this moment the IoTN can be considered as a scarce network, since only the $\frac{\rho_i(t)}{\beta}$ percent of the reliable links is connected.

Algorithm 1: Generator function to estimate the list of significant nodes (LSN) and the number of significant nodes ($\rho_i(t)$).

    Input: $i, LNR_i(t)$
    Output: $\rho_i(t), LSN_i(t)$
1 $j \Longleftarrow$ size of the $LNR_i(t)$ for the $MCU_i(t)$
2 Sort all $LNR_i(t)$ regarding their bandwidth in $S_{LNR_i}(t)$
3 Estimate $B_{thr} = \mu_i(t) - \sigma_i(t)$
4 Initialize the counter $label = 1$ and $LSN = \{0, 0, 0\}$
5 **for** $label \leftarrow 1$ **to** j **do**
6   Calculate $B_{LNR_k}(t)$: the bandwidth in Mbps of the k element in the $SLR_{LNR_k}(t)$
7   **if** $B_{LNR_k}(t) \geq B_{thr}$ *AND* $k \leq 3$ **then**
8         $LSN(k) = MCU_{label}(t)$
9         $k \Longleftarrow k + 1$
10    $LSN(k) = \{$ the three best links according B_{thr}, if possible $\}$
11    $\rho_i(t)$ is the number of elements $\neq 0$

4.3 Hilbert fractal scanning

The construction of the Hilbert space-filling fractal, under the paper's graphic interpretation framework, was reduced to the replacements of the smallest elements (symbols) or interior replacements. This path can be optimized by means of the correlation of the strength of the signal or the bandwidth of the link by Eq. (1):

$$\phi_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{1}$$

where ϕ_{xy} is the sample correlation coefficient, n is the stage 0 level of Hilbert curve (**Figure 2a**), and x_i and y_i are, respectively, the best and the second best individual bandwidths in a certain MCU_i indexed inside the IoTN as the ith element, while $\overline{x}$ and $\overline{y}$ are, respectively, the sample mean of the best and the second best individual bandwidths in a certain MCU_i. Thus, we can estimate the best link to a particular MCU_i.

All $\sum_{i=1}^{4^n} MCU_i(t)$ are distributed randomly giving as a result a shape as shown in **Figure 2b**. When γ is reordered as vector, it gives as a result the Hilbert curve indexing $\vec{\gamma}$ which contains the order of the $MCU_i(t)$ (see **Figure 2c**).

4.4 Initial topology

Once the Hilbert curve is defined as L-system, we adapt the production rules of the original work by Hilbert [4], who proposed an axiom with a $\mathcal{D}$ trajectory, while we propose to start with an $\mathcal{U}$ trajectory. Our proposal is based on the fact that most of the energy is concentrated in the nearest MCU, namely, at the right or left. In this way the production rules of the Hilbert curve are defined by:

- $\mathcal{U}$ is changed by $\mathcal{LUUR}$

- $\mathcal{L}$ by $\mathcal{ULLD}$

Figure 2.

1	4	5	6	59	60	61	64
2	3	8	7	58	57	62	63
15	14	9	10	55	56	51	50
16	13	12	11	54	53	52	49
17	18	31	32	33	34	47	48
20	19	30	29	36	35	46	45
21	24	25	28	37	40	41	44
22	23	26	27	38	39	42	43

(a)

64	2	3	61	60	6	7	57
9	55	54	12	13	51	50	16
17	47	46	20	21	43	42	24
40	26	27	37	36	30	31	33
32	34	35	29	28	38	39	25
41	23	22	44	45	19	18	48
49	15	14	52	53	11	10	56
8	58	59	5	4	62	63	1

(b)

(c)
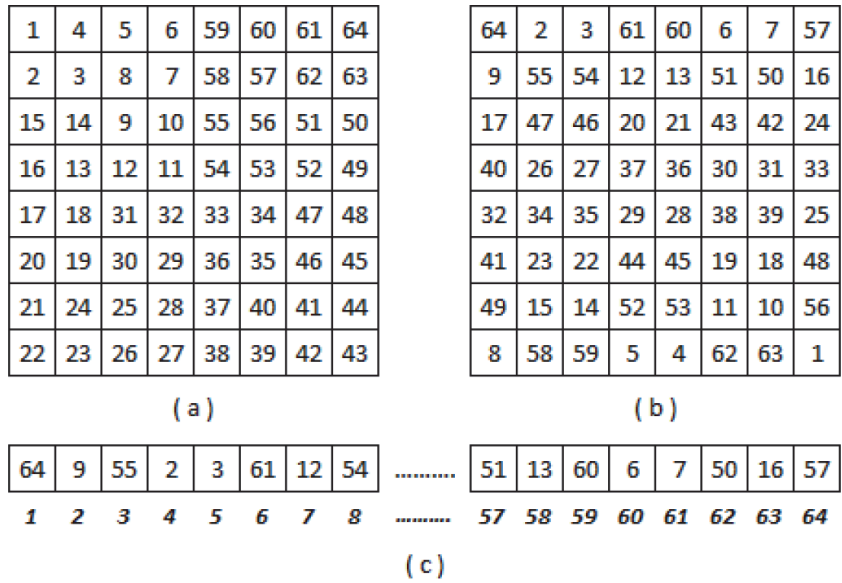
(a) First three stages of the Hilbert curve, (b) matrix of dispersion or γ of $\sum_{i=1}^{4^n}MCU_i(t)$ inside a convectional room, and (c) matrix $\vec{\gamma}$ ordered as a vector.

- $\mathcal{R}$ by $\mathcal{DRRU}$
- $\mathcal{D}$ by $\mathcal{RDDL}$

The Hilbert curve has the property of remaining in an area as long as possible before moving to a neighboring spatial region. Hence, the correlation between neighbors $MCU_i(t)$ is maximized, which is an important property in the optimization of any system. The higher the correlation for estimating the final topology, the more efficient the data parameter sharing.

5. Experimental results

In this section, first we approach the basic concepts of the network simulator (NS) and then use it to compare the most current SDN topologies, which helps us to identify the effectiveness of the proposed algorithm.

5.1 Network simulator

Network simulator is a software to simulate discrete events, and it was designed to aid in the research of telematic networks. NS provides support for the simulation of a multitude of protocols of the application layers (http, ftp, cbr, etc.), transport (TCP, UDP, RTP, SRM), unicast and multicast routing protocols, etc., both for networks wired as local or satellite non-wired and with complex topologies with a large number of traffic generators. NS began in 1989 as a variant to the existing REAL Network Simulator and has evolved substantially in recent years, having been developed by DARPA with the help of several network research institutions, such as LBL, Xerox PARC, UCB, USC/ISI, etc.

NS is basically an object-oriented simulator, written in C++, whose user interface is presented as an object-oriented Tcl language interpreter or, in other words, OTcl language. The simulator supports a hierarchy of classes written in C++, also
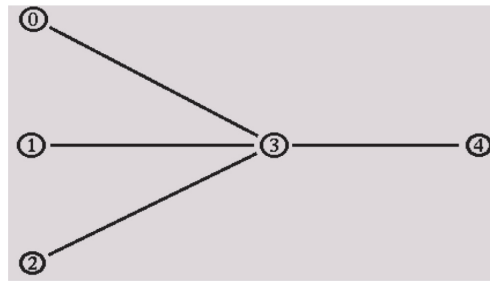
Figure 3.
Example of network topology.

called a compiled hierarchy, and another hierarchy of classes similar to the previous one but within the interpreter that is presented to the programmer in OTcl, and which is also known as an interpreted hierarchy. These two hierarchies are closely related to each other, so that, from the user's perspective, there is a one-to-one correspondence between one class in the interpreted hierarchy and one in the compiled hierarchy. The root of this hierarchy is a class called TclObject. When the user creates a simulator object from the interpreter, which usually starts a script, this object is created within the interpreter, and a close relationship is created with another identical object but within the compiled hierarchy. The interpreted class hierarchy is established automatically through methods defined within the class called TclClass. In addition, we can create other hierarchies to our liking in the scripts, written in OTcl, that are not split in the compiled hierarchy.

To create the topology, we just show a simple example for purposes of understanding the use of this simulator. For this example we use the connection depicted by **Figure 3**.

Then the nodes are created in the same way as in **Figure 3**, as follows:

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]


$ns duplex-link $n0 $n3 1Mb 100ms DropTail
$ns duplex-link $n1 $n3 1Mb 100ms DropTail
$ns duplex-link $n2 $n3 1Mb 100ms DropTail
$ns duplex-link $n3 $n4 1Mb 100ms DropTail
```

Then, traffic sources are attached to nodes n0, n1, and n2:

```
proc attach-expoo-traffic { node sink size burst idle rate } {
    #Get an instance of the simulator
    set ns [Simulator instance]

    #Create a UDP agent and attach it to the node
    set source [new Agent/UDP]
    $ns attach-agent $node $source

    #Create an Expoo traffic agent and set its configuration parameters
    set traffic [new Application/Traffic/Exponential]
    $traffic set packet-size $size
    $traffic set burst-time $burst
```

```
$traffic set idle-time $idle
$traffic set rate $rate
# Attach traffic source to the traffic generator
$traffic attach-agent $source
#Connect the source and the sink
$ns connect $source $sink
return $traffic
```

First, the procedure creates a traffic source and attaches it to the node. Then, it creates a Traffic/Expo object, sets its configuration parameters, and attaches it to the traffic source. Before, the source and sink have been connected. Now, the traffic sources (with different peak rates) are attached to n0, n1, and n2, and then they are connected to the three traffic sinks in node n4, which has been previously created:

```
set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]
$ns attach-agent $n4 $sink0
$ns attach-agent $n4 $sink1
$ns attach-agent $n4 $sink2

set source0 [attach-expoo-traffic $n0 $sink0 200 2s 1s 100k]
set source1 [attach-expoo-traffic $n1 $sink1 200 2s 1s 200k]
set source2 [attach-expoo-traffic $n2 $sink2 200 2s 1s 300k]
```

In this example, Agent/Loss Monitor objects are used as traffic sinks, since they store the amount of data received, which can be used to calculate the bandwidth. In the same way, we construct all the nodes and interconnections for the simulation of the next section.

5.2 Comparing SDN performance

Once NS is defined, we estimate the performance of this chapter by the simulation of the effectiveness of the project node and the quality of service. Moreover, the duration of the OFA was evaluated by the end of the delay (E2ED) and the packet delivery ratio (PDR). So we compared the performance of OFA to newer algorithms similar to the latest technology:

1. Reliable and energy-efficient routing (REER) proposed by Li et al. in [8]

2. Delay-energy trade-off in wireless sensor networks with reliable communication (DETR) proposed by Liu et al. in [9]

3. Reliable routing with distributed learning automaton (RRDLA) proposed by Mostafaei in [10]

We use the network parameters recommended by Zorzi and Rao in [11] and Karp and Kung in our experiments [12].

In this way, it is important to define E2ED as the time it takes a packet to reach its destination after it is generated by its source, it is important to set the time for the packet to be reached because its next delay and delivery delays will be targeted at PDR probability of a total of [13] knot; the life span may be reset to original packages before the expiration of [14]. According to this definition, the correlation between the maximal and minimal end-to-delinquency values of the η is

determined by the change between maximal and minimal packet delivery ratios. If the delay ends are stable, η will be equal to 0, and if the packet delivery rate is constant $\eta = \infty$, a delay will appear:

$$\eta(\varphi) = \frac{E2ED(\varphi)_{\max} - E2ED(\varphi)_{\min}}{PDR(\varphi)_{\max} - PDR(\varphi)_{\min}} \qquad (2)$$

where φ is the effect of node density or the requirements for reliability of QoS. According to the models proposed by Niu et al. [15], it is based on the experience described for Mostafaei in [10] and Zeng et al. in [16].

In this way, **Figure 4a** and **b** show the comparison between PDR and E2ED, respectively, in terms of node intensity and reliability. We can say that better algorithms are reached with the higher E2ED and the lower PDR. Thus, both algorithms get the best results and try to minimize delays when sending target data which can restore original packages. OFA is slightly better in both cases, but the differences are not higher than 1.97%. Additionally, the delay in RRDLA and OFA is noticeably changed by increasing the intensity of the network or QoS node. DETR and REER also have worst results than the two best algorithms.
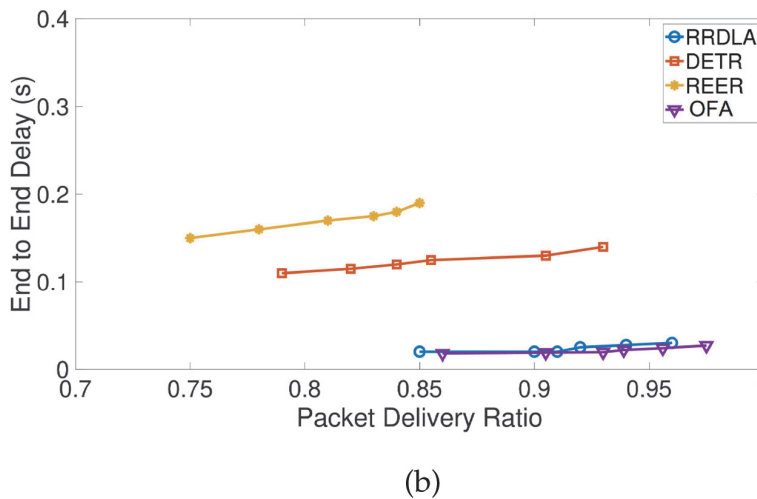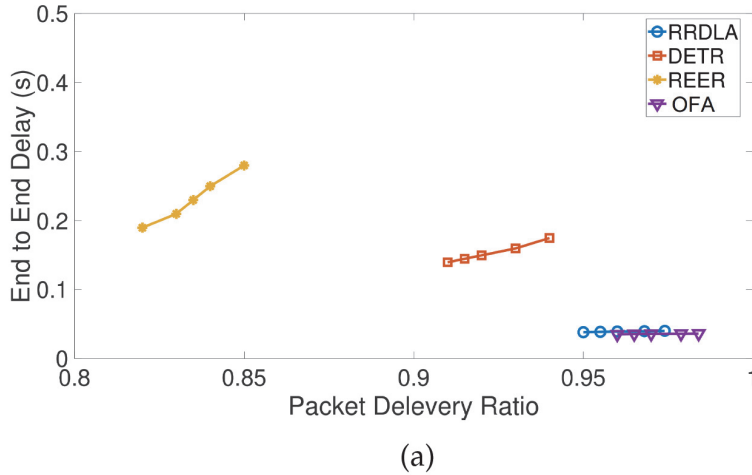

(a)

(b)

Figure 4.
The simulation of the impact of the node density and quality of service, case end-to-end delay vs. packet delivery ratio. (a) Node Density, and (b) Reliability Requirements.

Algorithm	η (node density)	η (quality of service)
RRDLA	*0.0833*	*0.0909*
DETR	1.1667	0.2143
REER	3.000	0.4000
OFA	**0.0386**	**0.0705**

Table 1.
η(φ) for the impact of node density and quality of service.

Table 1 shows $\eta(\varphi)$ for node intensity and service quality, both in the OFA or best results (bold) to 0 and RDDLA will have the second (emphasized).

If **Table 1** is analyzed, we can realize that OFA obtains the better result when node density and quality of service are compared. It is important to mention that using the regression techniques of a variable of E2ED on a PDR variable, we look for a function that is a good approximation of a point cloud (x_i, y_i); this regression algorithm reduces in an important way the time of processing. The simple linear regression model has the following expression:

$$E2ED = \alpha + \beta PDR + \varepsilon \qquad (3)$$

where α is the ordinate at the origin (the value that E2ED takes when PDR is 0), β is the slope of the line (and indicates how E2ED changes when increasing PDR in a unit), and ε is a variable that includes a large set of factors, each of which influences the response only in small magnitude, which we will call error. PDR and E2ED are random variables, so you cannot establish an exact linear relationship between them.

As a result, a package sent by OFA will spend less time to achieve its target, because the target node will increase the chance of recovering the original packet before its expiration.

6. Conclusions

Smart home technology, as part of IoT, is the future of residential-related technology which is designed to deliver and distribute services inside/outside the house via networked smart devices sharing information by means of a broadband Internet connection. In this chapter we developed an OFA for optimizing a central IoTN topology, increasing the range transmission of a given Wi-Fi network in an adaptive and dynamic way when sharing parameters, in order to achieve an energy-efficient IoT-based smart home.

With the IoTN topology proposed in this chapter, based on the L-system for the Hilbert space-filling fractal and focused on the efficient connection of the smart home devices or MCUs, the energy consumption of the connected devices is reduced. Most of this energy consumed by MCUs can be saved through inclusion of new nodes or devices connected in the same IoTN, since most of the information detected throughout the network is redundant due to geographically placed sensors.

Thus, our proposal optimizes the links among MCUs or smart home devices, since in a dense network we have 2016 possible links but with only 64 links we can share parameters of sensor, namely with only the 3.05% or we are reducing the 96.95% the number of links in the presented IoTN. In addition, we can manage scarce IoTNs in order to obtain more dense networks, since by means of a

backbone, many WAPs can be interconnected to form more complex or dense topologies, which can grow using higher levels of Hilbert fractal.

Finally, we propose a software-defined network (SDN) with strong mobility since it can be reconfigured depending on the amount of nodes; also we employ a target coverage because OFA only considers reliable links among smart home devices. In terms of reliability, our proposal can share parameters such as battery, radio, hardware, or operating systems. By itself, the quality of service is a challenge that guaranteed the level of service delivery to an IoTN; OFA takes less time to reach its destination after it is generated by its source increasing the probability that the destination smart home device can recover the original packet before the lifetime expires.

Acknowledgements

Author details

Jesús Jaime Moreno Escobar[1*†], Oswaldo Morales Matamoros[1†],
Hugo Quintana Espinosa[1†], Ricardo Tejeida Padilla[2†]
and Ana Gabriela Ramŕez Gutiérrez[3†]

1 Escuela Superior de Ingeniería Mecánica y Eléctrica, Instituto Politécnico Nacional, Mexico City, México

2 Escuela Superior de Turismo, Instituto Politécnico Nacional, Mexico City, México

3 Escuela de Administración de Instituciones, Universidad Panamericana, Guadalajara, México

*Address all correspondence to: jemoreno@esimez.mx

† All the authors contributed equally.

IntechOpen

References

[1] Mandelbrot BB. The Fractal Geometry of Nature. San Francisco, USA: Freeman; 1982. Chapter 7

[2] Sagan H. Space-Filling Curves. New York: Springer-Verlag; 1994

[3] Nicholas Rose J. Hilbert-Type Space-Filling Curves. Florida, USA: Bethune Cookman University; 2010

[4] Hilbert D. Über die stetige abbildung einer linie auf ein flächenstück. Mathematische Annalen. 1891;**38**(3): 459-460

[5] Prusinkiewicz P. Modeling and visualization of biological structures. In: Proceeding of Graphics Interface '93. Toronto, Canada; 1993. pp. 128-137

[6] Lindenmayer A. Mathematical models for cellular interaction in development, Parts I and II. Journal of Theoretical Biology. 1968;**18**:300-315

[7] Papert S. Mindstorms: Children, Computers, and Powerful Ideas. New York, USA: Basic Books; 1980

[8] Li X, Wang Y, Chen H, Chu X, Wu Y, Qi Y. Reliable and energy-efficient routing for static wireless ad hoc networks with unreliable links. IEEE Transactions on Parallel and Distributed Systems. 2009;**20**(10):1408-1421

[9] Liu Z, Dai L, Xue L, Guan X, Hua C. Reliability considered routing protocol in wireless sensor networks. In: Proceedings of the 30th Chinese Control Conference; 2011. pp. 5011-5016

[10] Mostafaei H. Energy-efficient algorithm for reliable routing of wireless sensor networks. IEEE Transactions on Industrial Electronics. 2019;**66**(7): 5567-5575

[11] Zorzi M, Rao RR. Geographic random forwarding (geraf) for ad hoc and sensor networks: Energy and latency performance. IEEE Transactions on Mobile Computing. 2003;**2**(4): 349-365

[12] Karp B, Kung HT. GPSR: Greedy perimeter stateless routing for wireless networks. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom '00); ACM: New York, NY, USA; 2000. pp. 243-254

[13] Liu J, Sheng M, Xu Y, Li J, Jiang X. End-to-end delay modeling in buffer-limited manets: A general theoretical framework. IEEE Transactions on Wireless Communications. 2016;**15**(1): 498-511

[14] Yang B, Chen Y, Cai Y, Jiang X. Packet delivery ratio/cost in manets with erasure coding and packet replication. IEEE Transactions on Vehicular Technology. 2015;**64**(5): 2062-2070

[15] Niu J, Cheng L, Gu Y, Shu L, Das SK. R3e: Reliable reactive routing enhancement for wireless sensor networks. IEEE Transactions on Industrial Informatics. 2014;**10**(1): 784-794

[16] Zeng K, Lou W, Yang J, Brown DR III. On throughput efficiency of geographic opportunistic routing in multihop wireless networks. Mobile Networks and Applications. 2007;**12**(5): 347-357