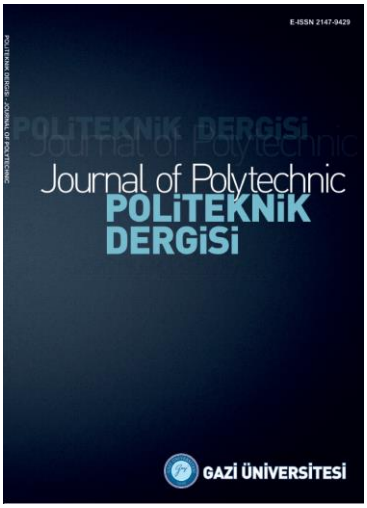

POLİTEKNİK DERGİSİ

JOURNAL of POLYTECHNIC

ISSN: 1302-0900 (PRINT), ISSN: 2147-9429 (ONLINE)

URL: http://dergipark.org.tr/politeknik

Applying toroidal k-ary grids for optimizing edge data centers

Uç veri merkezlerini optimize etmek için toroidal k-ary ızgaralarını uygulama

***Yazar(lar) (Author(s))**: Pedro Juan ROIG[1], Salvador ALCARAZ[2], Katja GILLY[3], Cristina BERNAD[4], Carlos JUIZ[5]*

ORCID[1]: 0000-0002-8391-8946

ORCID[2]: 0000-0003-3701-5583

ORCID[3]: 0000-0002-8985-0639

ORCID[4]: 0000-0001-9537-415X

ORCID[5]: 0000-0001-6517-5395

To cite to this article: Roig P.J. et al., "Applying Toroidal k-ary Grids for Optimizing Edge Data Centers", ***Journal of Polytechnic***, *(*): *, (*).

Bu makaleye şu şekilde atıfta bulunabilirsiniz: Roig P.J. et al., "Uç Veri Merkezlerini Optimize Etmek İçin Toroidal K-ary Izgaralarını Uygulama", ***Politeknik Dergisi***, *(*): *, (*).

Erişim linki (To link to this article): http://dergipark.org.tr/politeknik/archive

DOI: 10.2339/politeknik.1327964

Applying Toroidal k-ary Grids for Optimizing Edge Data Centers

Highlights

- *ACP*
- *Data Center Topology*
- *Edge Computing*
- *Toroidal k-ary Grid*
- *Toroidal Topologies*

Graphical Abstract

A dynamic framework is outlined based on toroidal k-ary grids so as to organize and optimize small data centers, allowing them to increase or decrease according to the current and predicted capacity of IoT-generated traffic flows.

	***k*=2**			***k*=3**	
	n=2	**n=3**	**n=4**	**n=2**	**n=3**
Number of active nodes	4	8	16	9	27
Degree of each node	2	3	4	4	6

Table. *Number of active nodes and degrees in the instances presented.*

Aim

Data centers need to get customized for the specific requirements of edge computing, such as a small number of physical servers and the ability to scale and unscale according to the traffic flows running at a given time. In this context, artificial intelligence plays a key part as it may anticipate when traffic throughput will increase or otherwise by scrutinizing current traffic whilst considering other factors like historical data and network baselines.

Design & Methodology

The steps of this study are first to introduce toroidal topologies, then to propose a model based on toroidal k-ary grids, after that to outline a machine learning layout to select a convenient topology, afterwards to propose a formal algebraic specificaciton of the model, and finally to verify that model.

Originality

This study proposes a system to swap the specific topology of a data center based on toroidal k-ary grids in a dynamic manner by means of applying machine learning to optimize the number of hosts on, which is a novel approach.

Findings

The dynamic framework based on toroidal k-grids proposed allows to achieve good performance whilst saving energy.

Conclusion

The dynamic framework proposed permits to have an optimal number of hosts up and running within a toroidal k-ary grid topology in a data center by means of applying machine learning techniques. Furthermore, a formal algebraic model of that framework has been specified and verified.

Declaration of Ethical Standards

The author(s) of this article declare that the materials and methods used in this study do not require ethical committee permission and/or legal-special permission.

Applying Toroidal k-ary Grids for Optimizing Edge Data Centers

Araştırma Makalesi / Research Article

Pedro Juan ROIG[1,2*], Salvador ALCARAZ[1], Katja GILLY[1], Cristina BERNAD[1], Carlos JUIZ[2]

[1]Department of Computer Engineering, Miguel Hernández University, Elche, Spain
[2]Department of Mathematics and Computer Science, University of the Balearic Islands, Palma de Mallorca, Spain

(Geliş/Received : 19.07.2023 ; Kabul/Accepted : 29.08.2023 ; Erken Görünüm/Early View : 13.10.2023)

ABSTRACT

IoT deployments are growing exponentially, leading to a huge increase in edge computing facilities. In order to cope with such a demand, data centers need to get customized for the specific requirements of edge computing, such as a small number of physical servers and the ability to scale and unscale according to the traffic flows running at a given time. In this context, artificial intelligence plays a key part as it may anticipate when traffic throughput will increase or otherwise by scrutinizing current traffic whilst considering other factors like historical data and network baselines. In this paper, a dynamic framework is outlined based on toroidal *k*-ary grids so as to organize and optimize small data centers, allowing them to increase or decrease according to the current and predicted capacity of IoT-generated traffic flows.

Keywords: ACP, Data Center Topology, Edge Computing, Toroidal *k*-ary Grid, Toroidal Topologies.

Uç Veri Merkezlerini Optimize Etmek İçin Toroidal k-ary Izgaralarını Uygulama

ÖZ

IoT konuşlandırmaları katlanarak artıyor ve uç bilgi işlem tesislerinde büyük bir artışa yol açıyor. Bu tür bir taleple başa çıkabilmek için veri merkezlerinin, az sayıda fiziksel sunucu ve belirli bir zamanda çalışan trafik akışlarına göre ölçekleme ve ölçeği kaldırma yeteneği gibi uç bilgi işlemin belirli gereksinimlerine göre özelleştirilmeleri gerekir. Bu bağlamda yapay zeka, geçmiş veriler ve ağ temelleri gibi diğer faktörleri göz önünde bulundurarak mevcut trafiği inceleyerek trafik hacminin ne zaman artacağını veya artacağını tahmin edebildiği için önemli bir rol oynar. Bu yazıda, küçük veri merkezlerini organize etmek ve optimize etmek için toroidal k-ary ızgaralarına dayanan dinamik bir çerçeve ana hatlarıyla açıklanmakta ve IoT tarafından üretilen trafik akışlarının mevcut ve tahmin edilen kapasitesine göre artmalarına veya azalmalarına izin verilmektedir.

Anahtar Kelimeler: ACP, Veri Merkezi Topolojisi, Edge Computing, Toroidal k-ary Grid, Toroidal Topolojiler.

1. INTRODUCTION

Internet of Things (IoT) deployments are exponentially ever-increasing, leading to the growth in edge computing facilities in order to cope with the traffic flows generated by all those IoT devices [1]. In this context, Artificial Intelligence (AI) advances make possible to significantly enhance the performance of edge computing so as to be able to deal with the huge levels of Big Data produced [2]. Hence, IoT and AI may work together in order to achieve optimal solutions through the use of Deep Learning (DL) tools applied to edge computing [3].

The partnership between IoT and AI has been successfully deployed in many scenarios, such as in the healthcare sector, where remote healthcare monitoring systems employ machine learning (ML) methods to help create analytic representations, whilst assisting in clinical decision support systems [4]. On the other hand, the use of IoT and AI in smart homes provides a wide range of possibilities in home automation whilst optimizing energy consumption [5].

Furthermore, some other common uses of IoT and AI pairing together are intelligent transport systems, which contribute to efficiently minimize traffic problems [6], smart cities, which lead to improve the lifestyle of citizens and public services [7], Industry 4.0, which brings process automation to contribute to the convergence of information technology and operation technology [8], or Retail 4.0, which enhances the integration of online and offline sales so as to create a personalized shopping experience [9].

In this sense, it is to be remarked that the advances in terms of deep learning are growing by the day, which allows to get the environments smarter on an ongoing basis [10]. One of the fields where such a progress is more evident is the interaction of physical elements and cyberspace items together, most commonly known as cyber-physical systems (CPS), along with the generation of digital twins (DT), which are virtual copies of physical items behaving the same way [11].

Hence, IoT may basically be seen as a concept referred to digitally interconnecting regular objects to the internet [12], leading to the interconnection of objects and people thanks to digital transformation [13]. IoT applications are

*Corresponding Author: Pedro Juan Roig
e-posta : proig@umh.es

exponentially growing in all areas, although the rise of CPS is to be highlighted for being a prominent one [14].

It is to be noted that CPS are based on recent advances of IoT technologies [15], which happen to extend the paradigm of distributed systems by implementing control cycles between physical and logical components as a way to satisfy common system objectives [16], which may well be applied to diverse domains like home automation, smart cities or Industry 4.0 [17].

In this context, the design of system architectures aimed at IoT environments is crucial in order to attain optimal routes to undertake exchanges among the physical and logical actors involved [18], where concerns about security and privacy must also be addressed in order to achieve not only IoT domains with appropriate designs but also assuring that those are safe and secure [19].

Therefore, the most relevant step in building up cyber-physical systems is to design an architecture offering fast forwarding capabilities [20], which may allow applications with real time constraints, as well as having a resilient infrastructure offering redundant paths so as to avoid single points of failure [21].

Taking into account the most prominent remote computing paradigm nowadays, it is to be said that cloud computing offers the higher computing power and its ubiquity [22], although it presents high levels of latency due to the distance where the cloud servers are located with respect to the end users [23], as well as the lack of bandwidth due to the use of Wide Area Network (WAN) technologies for data transmission to go from users to remote cloud servers, and the other way around [24].

However, other remote computing paradigms get the computer power closer to the end user, thus allowing for a reduction in latency and jitter values, whilst facilitating the use of Local Area Network (LAN) technologies for data transmission [25]. In this sense, fog computing locates the computing resources around the edge of the network [26], whilst edge computing situates such resources right on the edge of the network, or even within [27].

It is to be noted that the difference between fog and edge computing oftentimes gets blurred, although both of them enhance latency and bandwidth. Anyway, edge computing is recently receiving more attention by both industry and academia, even though there are still fog deployments being developed [28].

In any case, latency and bandwidth values obtained beat those provided by cloud computing, making the latter as a good backup alternative for more intense processing and also a storage solution [29]. Furthermore, either edge or fog computing offers a more sustainable alternative to cloud computing, as shorter distances account for power saving [30], which may also be green if renewable energies are used so as to provide power to it [31].

Therefore, it may be said that the main point to establish a cyber-physical system is to design an edge architecture aimed at IoT domains so as to achieve a resilient and sustainable solution [32]. This could be achieved by means of implementing redundant paths among the nodes being part of the edge data center [33], which may be done by proposing the appropriate designs [34].

In this sense, it is important to take into consideration some of the current challenges facing IoT deployments in order to properly empower all devices involved. The most relevant ones are those related to security & privacy, compatibility & standardization, storage & cloud, energy and communications [35].

To start with, the limited capabilities of IoT devices affects the development of advanced ways of encryption, thus leading to threaten data privacy. On the other hand, IoT devices are heterogeneous, hence the establishment of common grounds would make IoT work more efficiently [36].

Regarding storage and cloud, intelligent analytics mechanisms need to be adapted to deal with the vast amount of information being compiled by IoT environments, which leverages the adoption of decentralized architectures as edge computing or fog computing [37], with centralized frameworks as cloud computing act as backup solutions.

With respect to energy efficiency, this is crucial in order to minimize carbon emissions, whilst achieving great performance rates [38]. Likewise, communications are closely related to the former, as those need to make an efficient use of power and spectrum, whilst lowering interference levels low enough [39].

Taking all those challenges in mind, energy efficiency is the main pillar of this paper, where the design of an efficient architecture devoted to edge computing would result in a limited number of nodes, as the scope of edge environments is usually geographically limited, as opposed to cloud ones, which are usually global. This way, processing and storage would take place in the edge server, although cloud facilities may be used as a backup solution. Furthermore, efficiency would also be attained by means of achieving better performance due to the shorter distances among nodes provided by the design proposed, which will lead to lowering the carbon footprint.

As far as related work is concerned, edge computing environments are being developed swiftly, although some relevant works need to be mentioned, such as [40], which exposes an online mechanism called EdgeDR, aimed at attaining cost efficiency in edge demand response programs, whilst [41] exhibits a framework called S-Cache to cache popular services in edge cloud systems so as to reduce response times.

Likewise, [42] proposes a method called ECSD-SGD to accelerate decentralized training through error-compensated sparsification, whilst [43] depicts a clustering-based strategy called CLUB, focused on the analysis of urban mobility from the edge data centers point of view, whereas [44] reviews the energy use in data centers related to ICT equipment and physical infrastructure, along with how to improve sustainability.

Moreover, [45] presents federated learning as a way to train models across distributed clients in edge environments, thus enhancing collaborative learning whilst preserving data privacy, whereas [46] describes an efficient query processing engine for edge devices called FineQuery, which improves both latency and bandwidth values.

In this paper, a toroidal topology based on a dynamic set of *k*-ary grids is presented, that being based on the one exposed in [47]. The main feature of this frameset of toroidal grids is the fact that distances among nodes are minimized in each scenario, as it will be exposed in the next sections, thus lowering energy consumption to move among nodes, which brings a lower carbon footprint for the whole system [48].

The target of the model proposed is to optimize the organization of edge data centers in two ways, such as choosing the topology with the smallest number of nodes being able to deal with all computing necessities at a given time, as well as taking advantage of minimal distances among nodes due to the specific layout imposed, thus attaining more efficient operation times [49] and linking together carbon efficiency with the improvement of financial performance [50].

In other words, a framework composed of a set of toroidal *k*-ary grids is proposed in order for one of those to be the topology selected for the links connecting together the nodes in an edge data center. It is to be considered that the small size of such data centers results in a limited number of nodes, which allows to employ instances of toroidal *k*-ary grids with a small amount of vertices.

Furthermore, the topology *k*-ary grid selected may change dynamically according to traffic conditions, as an AI-based tool may find the most convenient one in terms of energy saving, thus putting in idle state the nodes not being used to deal with the incoming traffic flows. Therefore, a base case scenario with a minimum amount of nodes may be set up and running, which may get extended if needed by growing scenarios.

The novelty in this work is twofold, such as the concept of a toroidal *k*-ary grid topology and the outline of a frameset of instances of such a topology to dynamically adapt the size of an edge data center to the changing network traffic needs over time. On the one hand, it is to be noted that *k*-ary *n*-cube topologies and toroidal *k*-ary topologies share the same physical layout, that being a toroidal shape, although they differ in the logical organization of the nodes.

Regarding the former, two features may be quoted, such that node layout follows sequential patterns along the different dimensions of the torus, thus allowing to easily spot a given node in an ordered manner, even though not all nodes with only one discordant symbol are neighbors if $k \geq 4$, thus not being possible to make a direct move among all those nodes, which results in larger paths among them.

Hence, the latter comes up to sort out the second feature at the expense of the first one, this is, making a direct connection among all nodes with just one discordant symbol whilst messing the node layout so as to make it possible, although once a node has been spotted, redundant movements towards any other node may be easily found by following the different discordant symbols among them.

On the other hand, the frameset of toroidal *k*-ary grids proposed allows to implement an adaptive layout, such that different instances may dynamically change to better adapt to incoming traffic conditions. In this sense, five instances are defined within the frameset, whilst an AI tool will choose which of those is the one achieving the expected performance with the minimum number of nodes in an active state, which may rise or lower over time according to new predictions, thus contributing to save energy.

In order to predict the best instance to be used in advance, the AI tool will process current network traffic flows, as well as other additional factors, such as data related to historical registers, baseline measurements and network intelligence so as to indicate the most convenient instance to deal with the expected workloads.

With regards to the best uses of toroidal *k*-ary grids towards solving society problems, it is to be said that they may provide some interesting solutions in edge domains. For instance, dealing with location issues related to both spotting and tracking end devices on the move may take advantage of the patterns defined among neighboring nodes within this topology, as it will be shown later on, in order to detect the path being followed by the movement among nodes. Besides, finding redundant paths between any pair of nodes will also be pretty straightforward thanks to the visualization of those patterns, which may be used in resource migration operations as well.

Additionally, a further potential use would be to design a forwarding strategy based on swapping the discordant symbol between two neighboring nodes, as opposed to do it by employing IP or MAC addresses, which would require specific research about it. The point here would be the cost and the time used by the logical gates being embedded in a proper Application Specific Integrated Circuit (ASIC) to undertake the tasks of swapping a symbol compared to the cost and time of doing it through lookup tables of IP or MAC addresses, which ultimately depends on how those operations are implemented in hardware.

With respect to the best uses of the frameset of instances of toroidal *k*-ary grids towards solving society problems, it is to be noted that the aforementioned design is able to dynamically adapt to changing conditions in edge environments, such as non-constant traffic flows or high-demand traffic requirements, which generates workloads with different characteristics over time, hence needing a different amount of nodes to get them processed properly.

Hence, the frameset proposed may be able to choose the appropriate topology so as to deal with the incoming

workloads by using the topology instance including the minimum necessary amount of nodes up and running. Such a number may be recalculated on a regular basis in order to check whether it is necessary to increase or decrease it, which may lead to the selection of a more convenient instance to adapt to the new circumstances, thus letting the edge data center work in an optimal way.

The rest of the paper is organized as follows: Section 2 lists some of the main toroidal topologies, Section 3 exposes toroidal binary grid topologies, Section 4 explains toroidal *k*-ary topologies, Section 5 exhibits the model proposed, Section 6 presents a formal algebraic specification of the model, Section 7 describes the verification of the model and Section 8 draws the final conclusions.

2. TYPES OF TOROIDAL TOPOLOGIES

First of all, a toroidal topology may be defined as an *n*-dimensional matrix where each node has a link to both its predecessor and its successor for each dimension, whilst nodes located at the edges of a given line within a particular dimension get connected together by means of a wraparound link.

There are two variables defining a toroidal shape, such as *k* and *n*, whose values are natural numbers, where the former establishes the alphabet being used, which accounts for the values going from 0 all the way to *k*-1, whilst the meaning of the latter depends on the design considered, even though it usually has to do with the number of dimensions involved. Anyway, the overall amount of nodes in a toroidal topology is given by k^n. In this sense, Figure 1 depicts a generic toroidal instance where *k*=3, thus showing 3 nodes per dimension, and *n*=2, thus exhibiting 2 dimensions, hence resulting in a total of $3^2 = 9$ nodes overall.

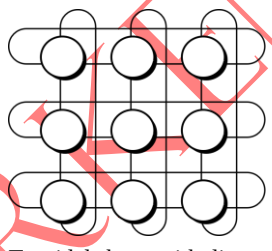

Figure 1. Toroidal shape with dimensions 3×3

One of the most well known applications of toroidal shapes are de Bruijn sequences, which involve strings of k^n length containing all available patterns made with an alphabet *k* and having a length *n*, with the particularity of including such patterns precisely once. De Bruijn sequences are obviously unidimensional as the whole shape and its patterns are strings, although they may be extended to further dimensions, thus generating de Bruijn tori for bidimensional shapes and patterns, no matter whether they are square or rectangular, or even de Bruijn hypertori for higher dimensions.

It is to be noted that each node belonging to a de Bruijn sequence contains a unique pattern out of the set of the available ones (k^n), whereas every node being part of a de Bruijn tori or hypertori contains just a single symbol within the *k* alphabet, such that each given node is appointed as the handle of a unique pattern from the set of the available ones (k^n). For instance, in the tridimensional case, the handle is the node located on the left, top, front corner, and from there on, its associated pattern is formed with the corresponding nodes staying rightwards, downwards, backwards out of the handle.

In this context, Figure 2 depicts the shape layout and mapping to achieve all available patterns exactly once for a square de Bruijn torus 4 × 4 with square patterns 2 × 2 using a binary alphabet, which is expressed as $(4,4;2,2)_2$. As a side note, it is to be said that this is the smallest possible de Bruijn torus, where the disposition of ones take the shape of a Brigid's cross. Actually, two different dispositions are feasible, as the matrix shown in Figure 2 is called the clockwise version, whereas its transposed matrix is named the counterclockwise one.

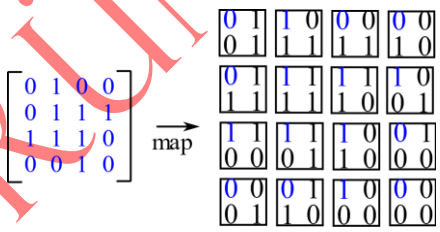

Figure 2. Clockwise disposition of $(4,4;2,2)_2$ and its mapping

A sufficient condition for a de Bruijn hypertori to exist is given by expression (1), where the dimensions of the shape are defined by variables r_i and those of the patterns are given by variables m_i, whilst variable *n* represents the amount of dimensions involved and variable *k* stands for the alphabet being used *k*. Obviously, it also applies to *n*=2, and even to *n*=1 if r_1 and m_1 are considered as the lengths of shape and patterns [51].

$$\prod_{i=1}^{n} r_i = k^{\prod_{i=1}^{n} m_i} \quad (1)$$

As stated above, $(4,4;2,2)_2$ represents a square toroidal matrix (4×4) with all available square patterns (2×2) with a binary alphabet (*k*=2). Additionally, Figure 3 exhibits some other representative instances of de Bruijn tori, where all of them meet the requirement exposed in (1). To start with, $(16,32;3,3)_2$ stands for a rectangular toroidal matrix (16×32) with all available square patterns (3×3) with a binary alphabet (*k*=2). Also, $(8,8;3,2)_2$ stands for a square toroidal matrix (8×8) with all available rectangular patterns (3×2) with a binary alphabet (*k*=2). Besides, $(4,16;3,2)_2$ stands for a rectangular toroidal matrix (4×16) with all available rectangular patterns (3×2) with a binary alphabet (*k*=2).

Therefore, it is clear that de Bruijn tori may exist as long as expression (1) is met, where both toroidal shapes (r_i) and toroidal patterns (m_i), with any alphabet k, may be either square or rectangular.

$(16,32;3,3)_2$	$(8,8;3,2)_2$	$(4,16;3,2)_2$	$(9,9;2,2)_3$	$(16,16;2,2)_4$
00000010111001011100101110010111	00000101	0000011101010011	000100010	0010001030203020
00010011111101001101101010000110	00100111	0011001100100110	001221221	0001020301000203
00000010111001011100101110010111	00110110	0011001010111011	111121211	0111011131213121
11101100000001011001001010111001	01000001	1111100100110001	112002002	1011121311101213
00000010111001011100101110010111	11111010		221201021	0010001030203020
00010011111101001101101010000110	11011000		110210120	2021222321202223
00110001110101101111100010100100	11001001		002012102	0111011131213121
10001010011011010100001100011111	10111110		222022202	3031323331303233
11111101000110100011010001101000			220110110	0313031333233323
11101100000001011001001010111001				1011121311101213
10101000010011110110000100111101				0212021232223222
01000110101000011000111111010011				0001020301000203
11111101000110100011010001101000				0313031333233323
11101100000001011001001010111001				2021222321202223
01100100100000111010110111110001				0212021232223222
11011111001110000001011001001010				3031323331303233

Figure 3. Some representative instances of de Bruijn tori

Moreover, $(9,9;2,2)_3$ stands for a square toroidal matrix (9×9) with all available square patterns (2×2) with a ternary alphabet (k=3). Furthermore, $(16,16;2,2)_4$ stands for a square toroidal matrix (16×16) with all available square patterns (2×2) with a quaternary alphabet (k=4).

Another well known application of toroidal shapes are k-ary n-cubes, which generalize the concept of n-hypercube to more than two nodes per dimension. In this case, n represents both the number of dimensions of the shape and the number of symbols identifying each node, where each of them refers to a given dimension. Besides, k stands for the k-ary alphabet being used. Therefore, n values range from 0 to n-1, whilst k values go from 0 to k-1, as shown in Figure 4, where k=3 and n=2.

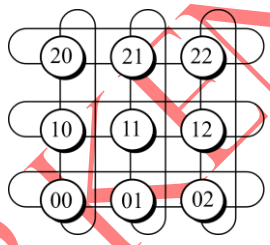

Figure 4. k-ary n-cube with k=3 and n=2

In that particular case, each node is represented by 2 symbols, as n=2, where it is easy to see that the least significant symbol is associated with the horizontal component, whereas the most significant one is assigned to the vertical one, although it could be any other way in other implementations or in higher dimensions. It is to be noted that there is an initial reference node, situated in a corner (in this case, it is the left bottom corner), whose symbols are all set to zero, and from there on, the successive nodes in any dimension grow sequentially, such that the least significant symbol grows horizontalwise whilst the most significant one does verticalwise.

Therefore, every node gets identified by a string of digits d_i, where i stands for its associated dimension, following this structure: $d_0...d_i...d_{n-1}$ where each of those symbols may get a value within the k-ary alphabet, this is, from 0 all the way to k-1. Furthermore, as each node is attached to its predecessor and its successor in each dimension i, then all neighboring nodes in such a dimension i meet expression (2), stating that all possible values range from 0 to k-1 [52].

$$d_i \pm 1 \ mod \ k \tag{2}$$

The layout of k-ary n-cube shapes are pretty straightforward because of the sequential arrangement of nodes. Furthermore, it may be appreciated in Figure 4 shown above or in Figure 5 shown below that all nodes with only one discordant symbol are neighbors, which allows for an easy way to move from one another by just taking the link representing the change of that symbol. However, such a feature does not apply to k-ary n-cubes where $k \geq 4$, as it may be seen in Figure 6.

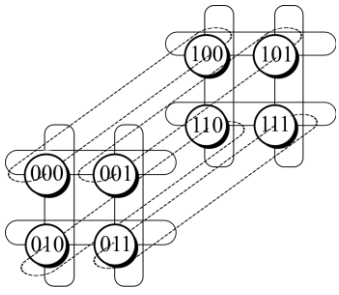

Figure 5. k-ary n-cube with k=2 and n=3

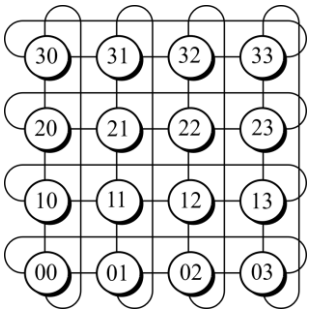

Figure 6. *k*-ary *n*-cube with *k*=4 and *n*=2

Hence, this proposition leads to another application of toroidal shapes by applying the condition that all nodes with only a discordant symbol must be neighbors, which may be branded as toroidal *k*-ary grid. In this case, it is to be considered that each node is labeled with *n* symbols, as stated in the *k*-ary *n*-cube. Furthermore, if the alphabet being used is the binary one, then the shape may be rebranded as toroidal binary grid.

The fact that all nodes with exactly a discordant symbol are neighbors minimizes distances among nodes, as the distance between any pair of nodes is established by the number of discordant symbols between them, because there is a direct link to deal with each of those. Also, if more than one discordant symbol is present, then redundant paths are available, as any order may be established to pass through each of those direct links.

Additionally, it is to be remarked that toroidal *k*-ary grids could be exhibited as toroidal bidimensional graphs when n is even, with every node having a degree *n*. When arranging all the nodes in that way, horizontal and vertical patterns related to discordant symbols being in common by every node may be spotted all over the graph, thus showing them between any pair of neighboring nodes. Furthermore, it is to be said that an important characteristic of such graphs is that links do not cross each other. In this sense, Figures 8, 9 and 13 will depict examples of toroidal bidimensional graphs, as *n* is even in those cases.

On the other hand, if *n* is odd, then it is not possible to obtain a toroidal bidimensional graph. However, this could be sorted out in two ways, such as either constructing a non-toroidal bidimensional grid, or otherwise, building up a toroidal *n*-dimensional grid.

Regarding the former, the patterns are created by following the number of zeros contained in the node identifiers, as opposed to the discordant symbols, although links do cross each other, whilst with respect to the latter, the patterns do contain the discordant symbols, even though they need to be arranged by taking *n* dimensions in order for links not to cross each other and to achieve a toroidal shape.

In this sense, Figures 10 and 14 will exhibit examples of non-toroidal bidimensional graphs, whereas Figures 11 and 15 will portray instances of toroidal *n*-dimensional graphs, because *n* is odd in these cases.

3. TOROIDAL BINARY GRID

As binary alphabets (*k*=2) allow up to 2 values per bit, the geometric shapes associated to them are the *n*-hypercubes, where *n* stands for the number of dimensions involved. Considering that one of the nodes is taken as the reference, where all its bits are set to zero, then any other node may have some bits set to zero if their values are the same as the reference one, or otherwise, such bits are set to one. This may be graphically seen in Figure 7, where *n*-hypercubes with the values of *n*={2,3,4} are depicted.

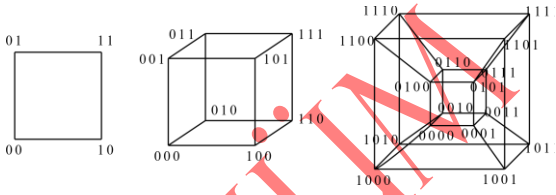

Figure 7. *n*-hypercubes for *n*=2, *n*=3 and *n*=4

Focusing on the case where *n* is even, the $k^n = 2^n$ nodes need to be arranged along a flat surface so as to obtain a toroidal bidimensional binary grid. Looking at Figure 8, where *n*=2, it is straightforward to see that patterns are spotted for rows and columns, where fixed values for determined bits are exhibited along with non-fixed values for some other bits, where the former are denoted as the corresponding values 0 or 1, whilst the latter are described as *x*. An additional advantage of this layout is that moving from one node to another may easily be done by following any of the redundant paths indicated by the patterns.

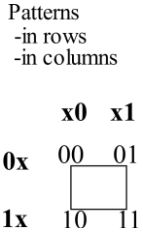

Figure 8. Toroidal bidimensional binary grid for *n*=2

Likewise, Figure 9 presents the toroidal bidimensional binary grid for *n*=4, where nodes have been properly arranged to achieve such a shape. In this case, patterns are also spotted for rows and columns, with fixed values for some bits, those shown as 0 or 1, whilst non-fixed values are done as *x*. As stated above, moving from a node to another one may easily be undertaken by means of the redundant paths exposed by the patterns.

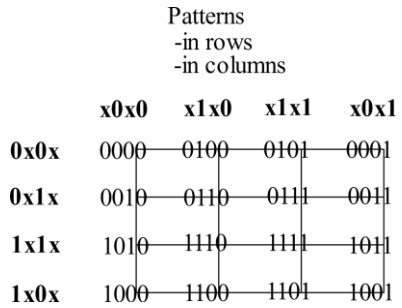

Figure 9. Toroidal bidimensional binary grid for n=4

Centering on the case where n is odd, the $k^n = 2^n$ nodes may be located along a flat surface, even though some links among them get crossed over, thus leading to attain a non-toroidal bidimensional binary diagram. Spotting at Figure 10, where n=3, it is to be said that patterns involve the number of zeros of a node identifier, which do not achieve a toroidal shape. However, redundant paths to move from one node to another could be found out by looking at the appropriate links within the diagram.

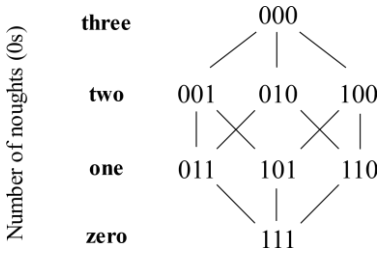

Figure 10. Non-toroidal bidimensional binary diagram for n=3

An alternative design for the case where n is odd is achieved if nodes are arranged in a toroidal n-dimensional binary grid, where patterns extend those found in bidimensional grids. For instance, Figure 11 exhibits a toroidal cubic binary grid which takes up to 3 dimensions (3D), with different layers shown separately in order to fit a 3D design into 2D so as to stick into this page.

It is to be noted that patterns involve rows, columns and layers, where the corresponding patterns are shown in each row, column and layer, respectively. Besides, the toroidal nature of the grid is preserved in 3D, whereas redundant paths to move from node to node could be observed by looking into the 3D-grid.

In other words, the front layer of the toroidal tridimensional binary grid is the one on the left of the picture, whereas the back layer is the one on the right, where both layers are linked together by regular links and wraparound links so as to join together the same corners in both layers, in the same way as corners within the same layer are linked together with rows and columns. This way, it is possible to move along any pair of vertices within the shape by just swapping the discordant bits, where each bit position is associated with a particular dimension of that shape.

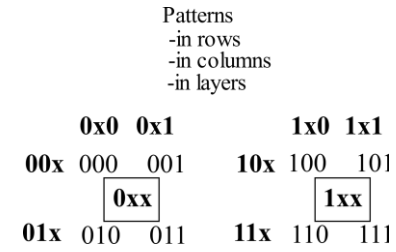

Figure 11. Toroidal tridimensional binary grid for n=3

4. TOROIDAL K-ARY GRID

Extending the binary case to k-ary ones, the geometric shape associated to them are the k-ary n-cubes, which have been described above. In this case, up to k values per dimension are allowed, those ranging from 0 all the way to k-1 in a sequential manner. Figure 12 shows two examples, where an instance for k=3 and n=2 is depicted, along with another one for k=3 and n=3. However, as exposed above, k-ary n-cubes do not meet the requirement of linking together all nodes with just one discordant bit when $k \geq 4$, thus nodes need to be rearranged as a toroidal k-ary grid to achieve so with any value of k.

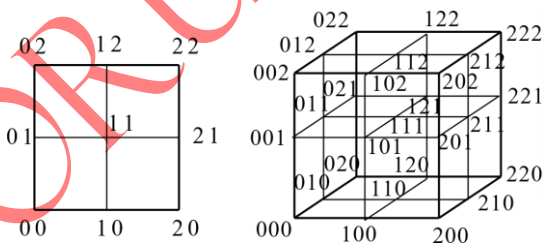

Figure 12. k-ary n-cube for k=3 and n=2 (left), along with k=3 and n=3 (right)

As toroidal k-ary grids extend the binary scenario, then the distinction between the case where n is even or odd also applies herein. Hence, sticking to the case where n is even, the k^n nodes are to be arranged along a flat surface in order to obtain a toroidal bidimensional k-ary grid, where patterns are also spotted for rows and columns. Figure 13 exhibits the case where k=3 and n=2, where the same implications described in its binary counterpart applies herein.

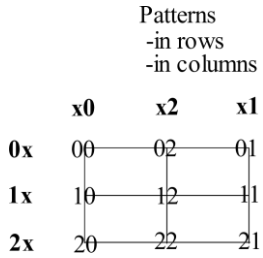

Figure 13. Toroidal bidimensional k-ary grid (k=3) for n=2

Besides, taking the case where n is odd, the same two alternative cases available for its binary counterpart applies. This way, a non-toroidal bidimensional k-ary diagram offers patterns based on the number of zeros,

even though some links get crossed over, whereas a toroidal *n*-dimensional *k*-ary grid displays patterns being an extension of the bidimensional schemes, such as rows, columns, layers, and so on, although links do not cross over due to the *n*-dimensional nature of the lattice.

Focusing on the former, Figure 14 depicts the particular case for *n*=3, meaning a non-toroidal bidimensional *k*-ary diagram (*k*=3), which happens to take the form of a crown, although other forms may be obtained for diverse values of *k* and *n*.

Centering on the latter, Figure 15 also exposes the specific case for n=3, meaning a toroidal tridimensional *k*-ary grid (*k*=3), where the different layers are split up. Moreover, the different patterns present in every row, column and layer are spotted.

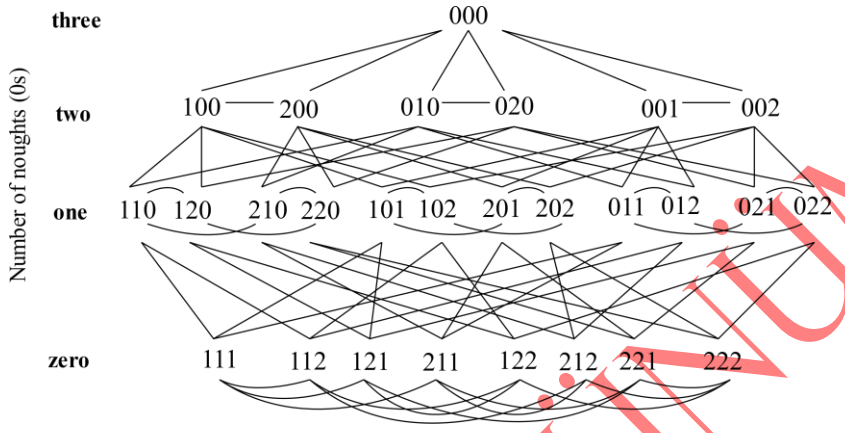

Figure 14. Non-toroidal bidimensional *k*-ary diagram (*k*=3) for *n*=3

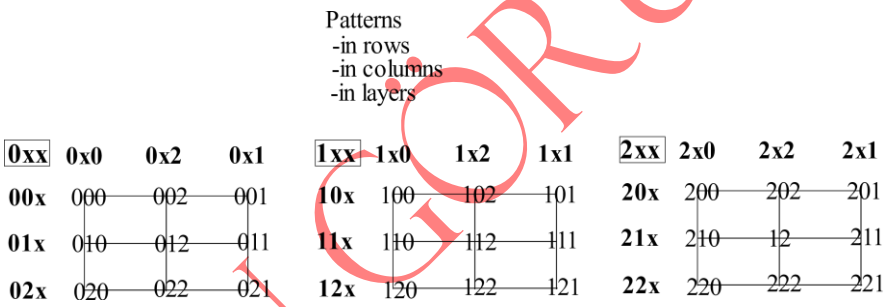

Figure 15. Toroidal tridimensional *k*-ary grid (*k*=3) for *n*=3

5. MODEL PROPOSED BASED ON TOROIDAL K-ARY GRIDS

The model proposed in this paper consists of a framework of diverse toroidal *k*-ary grids, which will integrate the five instances presented above. It is to be reminded that the main feature of toroidal *k*-ary grids is distance minimization among any pair of nodes, as all neighboring nodes have just one discordant symbol, meaning that the rest of their symbols must be concordant. In other words, if a node is identified by *n* symbols, then any couple of neighboring nodes shares *n*-1 matching symbols, whilst having 1 mismatching digit. This way, there will always be a path between two nodes composed by a number of hops equal to the amount of mismatching symbols between them, which will happen to be the shortest path between them. Furthermore, if the number of mismatching symbols is greater than one, then redundant paths will be feasible by combining the hops associated to each mismatching symbol.

It is to be noted that the aim of this study is to optimize the size of an edge data center, which usually occurs to contain a small to medium number of nodes because of the scope of such facilities. For this reason, the five instances shown above may be sufficient to deal with the different scenarios of throughput happening in an ordinary edge data center. In order to summarize such instances, Table 1 exposes the values of *k* and *n* for each one, along with the corresponding number of active nodes and the degree of each node, that being its number of neighbors in a toroidal environment. It is to be reminded that the former is given by the expression k^n, whilst the latter is done by the expression $n(k\text{-}1)$.

Table 1. Number of active nodes and degrees in the instances presented

	k=2			*k*=3	
	n=2	*n*=3	*n*=4	*n*=2	*n*=3
Number of active nodes	4	8	16	9	27
Degree of each node	2	3	4	4	6

By looking at the results shown, the scenario with the smallest number of nodes is k=2 and n=2, with just 4 nodes. This layout is appointed as the case base scenario of the model, as it implies the lowest number of active nodes, allowing the rest of them to go idle. This should be the default configuration of the model, which ought to be the one being used whenever possible, also implying a lower amount of carbon emissions.

In other words, whenever the traffic throughput may be dealt with just 4 nodes, which accounts for the active nodes for k=2 and n=2, then that layout must be up and running. However, at some point, traffic throughput may increase, which will lead to activate more nodes to cope with it. When such a case arises, then there are two diverse strategic lines to be followed, where one is bound for higher levels of traffic and the other one is done for much higher levels. On the one hand, regarding the former, it includes the scenarios where the value of n grows whilst k remains immutable, and that is why it is labeled as n-line. On the other hand, with respect to the latter, it contains the scenarios where the value of k gets higher, and in turn, n rises, which leads to be branded as k-line.

Focusing on the n-line, the first scenario involved is k=2 and n=3, leading to have 8 active nodes, whilst the second one is k=2 and n=4, presenting 16 active nodes. Centering on the k-line, the first step requires k=3 and n=2, getting 9 active nodes, whereas the second one does k=3 and n=3, obtaining 27 active nodes. Those results exhibit that the first step hardly makes a difference in both cases, although the second one indeed does, and that is why the n-line fits better for dealing with higher traffic, whereas the k-line suits better for managing much higher traffic.

Therefore, the framework of toroidal k-ary grids proposed may include the case base scenario (k=2, n=2), as well as both scenarios attached to the n-line (k=2, n=3 and k=2, n=4) and both scenarios assigned to the k-line (k=3, n=2 and k=3, n=3), accounting for an overall amount of 5 different scenarios, each of them having been presented above. Hence, as the scenario with the highest number of nodes is the last one, having a tally of 27 nodes, then such a number will need to be the minimal amount of necessary nodes to build up the framework presented, leaving aside the extra nodes being considered for backup purposes.

Additionally, the scenario being implemented may dynamically change to better adapt to traffic conditions. In this context, regarding the links among nodes, it is to be noted that each particular scenario has its links among specific nodes, such that the links for a given topology are not the same as those for another one. Hence, if a change of topology occurs, then the number of active and idle nodes will change to adapt to that new layout, whereas the links among nodes corresponding to the new topology will get up, whilst the links among nodes associated to the old topology will get down.

In order to better visualize the different scenarios proposed for the n-line and the k-line, Figure 16 depicts the number of running nodes in the five layouts established, taking into account that the initial stage in both lines is common, as it is the same in both cases, namely, it contains 4 running nodes (k=2, n=2). This case scenario is labeled as 0 in the horizontal axis of the picture.

Furthermore, it is shown that the first stage in both lines, which is branded as 1 in the horizontal axis, is quite similar, as the n-line stands for 8 running nodes (k=2, n=3) and the k-line portrays 9 running nodes (k=3, n=2). However, the second stage, which is tagged as 2 in the horizontal axis of the picture, does make a difference between both lines because the n-line presents 16 running nodes (k=2, n=4), whereas the k-line takes 27 running nodes (k=3, n=3).

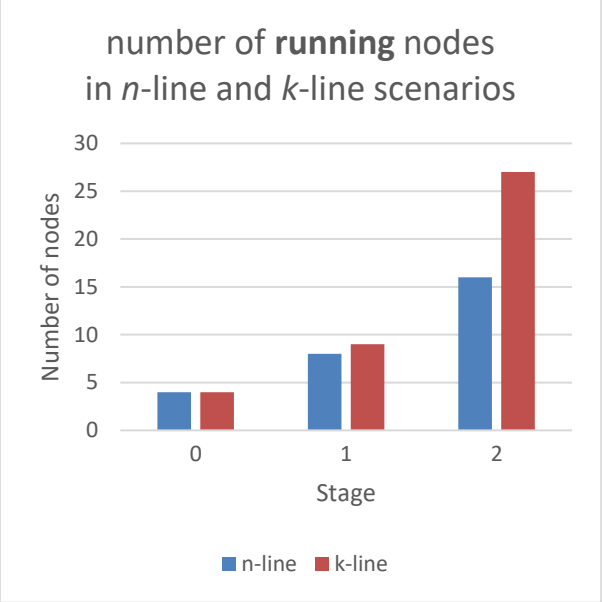

Figure 16. Number of running nodes in the different scenarios

On the other hand, Figure 17 exhibits the number of idle nodes in the five layouts proposed, where the overall amount of nodes considered is just the number of nodes of the most crowded scenario, which accounts for $k^n=3^3=27$. This way, 27 nodes stands for the sum of running node and idle nodes in any scenario. It could be argued that a certain amount of backup nodes should also be included for redundancy purposes, although this fact would just account for adding an offset to all values shown in the picture.

Anyway, as in the previous figure, the initial stage in both lines (k=2, n=2) contains only 4 running nodes, hence the amount of idle nodes is 27-4=23. Moreover, the first stage exhibits a similar amount of running nodes in both lines, as the quantity of idle nodes is 27-8=19 for the n-line (k=2, n=3), whilst it is 27-9=18 for the k-line (k=3, n=2). Nonetheless, the second stage exposes a significant difference in running nodes between both lines, and as a consequence, the total number of idles nodes in both lines differ, because the n-line accounts for 27-16=11 (k=2,

n=4), whereas its counterpart in the k-line yields 27-27=0 (k=3, n=3).

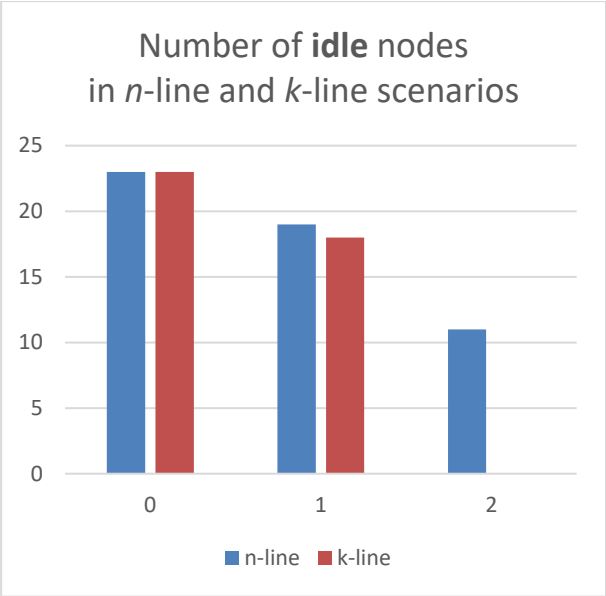

Figure 17. Number of idle nodes in the different scenarios

After having presented the components of the model based on a framework of toroidal k-ary grids, which induces lower carbon emissions due to the minimization of distances among nodes, it is time to discuss the way to find out the most convenient scenario in terms of balancing energy consumption and performance for each situation regarding different traffic conditions arising in the network at any given time.

The key point to obtain a forecast as accurate as possible is to take into consideration not only the current network conditions but also additional factors such as historical data, recent baseline measurements or data related to network intelligence from other sources. Additionally, as network conditions and intelligence information may well vary in a dynamic fashion, so the forecast must, hence the best way to dynamically establish the scenario fitting the most with some particular conditions at any given time is the application of AI tools in order to deal with all those data in a swift and precise manner.

The AI tool involved may well be CNN, as they provide more accurate results when solving many real-world problems than other AI counterparts. Basically, the idea is to compose an expression with determined coefficients being associated to current network conditions (α), historical data (β), recent baseline measurements (γ) and network intelligence (δ), as exposed above. Additionally, such coefficients should be able to dynamically adapt their values to reflect the potential changes in any of those conditions by increasing or decreasing their values on a per-situation basis.

Therefore, the instance of AI tool implemented will have six input values, such as the current values for k and n, along with the four aforementioned parameters, which are all related to rates of both bandwidth and latency. Basically, α may account for those values at present, β may do for those average values within certain time interval in the past, γ may go for those values on a regular basis and δ may stand for external information affecting those values in the near future. On the other hand, there will be two output parameters, such as the next values for k and n.

At this point, it may be interesting to state the relationship between Data Science (DS) and AI, as their bounds are not crystal clear. In this sense, the former refers to the scientific approach used to extract insights out of data by means of patterns, whilst the latter does to the tools and techniques to do it by means of first learning from current data, and in turn, predict future data. In other words, AI tries to obtain models out of making generalizations from data, whilst statistics and data analysis focuses on quantification of such data by try to infer and fit the parameters and errors out of them. Figure 18 exhibits the relation among data, models and statistical inference in a simple manner [53].

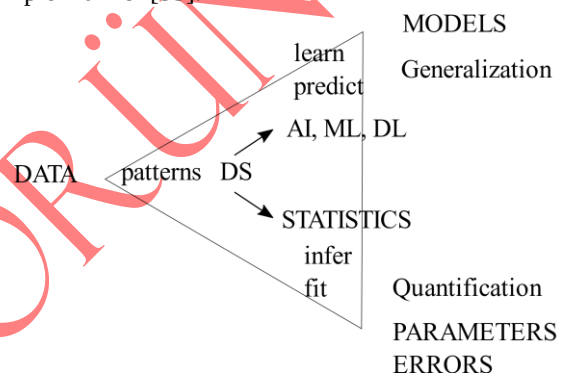

Figure 18. Relation among data, models and statistical inference

Regarding AI, it may also be defined as a program being able to sense, reason, act and adapt. On the other hand, machine learning (ML) may be seen as a subset of AI where algorithms improve their performance over the time whilst they get exposed to more data [54]. Moreover, deep learning (DL) may be considered as a subset of ML where multilayer neural networks learn out of vast amounts of data [55]. Furthermore, DL provides interesting features such as a universal learning approach, robustness, generalization and scalability. Additionally, the most commonly instances of DL are recursive neural networks (RvNN), recurrent neural networks (RNN) and convolutional neural networks (CNN), which is the most famous because of automatically recognizing the most relevant features without any human supervision [56].

Focusing on CNN, they are composed by three essential types of layers. The first one is the convolutional layer, where a matricial filter is applied by means of convolutional operators on the input so as to transform it into a matrix called feature map. The second one is the pooling layer, where an average or a max-pool operation is applied across the whole feature map with a given stride in order to lower the complexity of the feature map.

The third one is the fully connected layer, which reconnects the processed portions by means of diverse activation functions to find the highest probability values to better assign the output [57].

Hence, a CNN may be divided into three parts, such as an input layer, where the initial values feed the structure, then an architecture formed by a bunch of hidden layers, where usually convolutional and pooling layers get stacked together in pairs at the head of the architecture, whilst the fully connected layers are stacked with one another at the tail of the architecture, and eventually the output layer, where the final results are get out of the CNN [58].

A draft with the structure of the diverse layers in a CNN is exhibited in Figure 19, where items in a given column within the hidden layers have a full mesh connectivity with both elements in the predecessor and the successor columns [59]. It may be appreciated that convolutional and pooling layers are stacked in pairs on the left, whereas fully connected layers are just on the right [60].

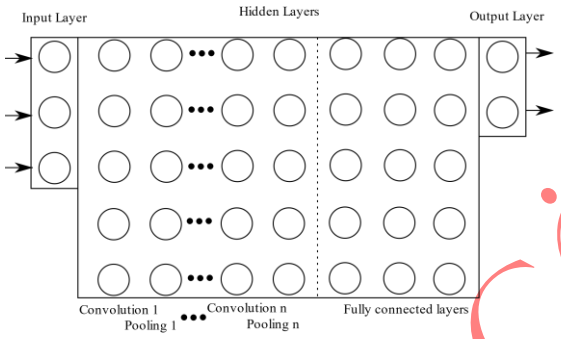

Figure 19. Structure of the layers in a CNN

Setting the focus on the framework of toroidal topologies proposed, the input is composed by the current *k* and *n* values, as well as the current traffic throughput, the historical data, the recent baseline measurements and network intelligence information. Regarding the hidden layers, a number of stacked pairs of convolutional and pooling layers are in charge of making the appropriate calculations, followed by a number of fully connected layers to achieve the final results. In this sense, it is to be noted that the higher the number of layers, the more precision is obtained, although the more processing time is needed. A possible setup could be 3 stacked pairs of layers so as to properly combine all inputs and 3 fully connected layers, as this is half of the number of inputs. With respect to the output layer, the new values of *k* and *n* will be exposed as the result of all calculations carried out within the hidden layers.

Therefore, CNN computations will eventually decide which of the five topologies proposed is the more convenient, in a way that an outstanding performance is attained, thus having a high enough number of nodes, whilst meeting energy-saving policies, hence having as many nodes in idle state as possible [61]. Additionally, when the input conditions change in a significative way, then CNN will undertake calculations in order to adapt the values of *k* and *n* to the new input conditions.

Although CNN seems to be a consistent option among ANN regarding the obtention of accurate forecasting models [62], it is to be mentioned that diverse authors discuss about the convenience of using CNN for processing historical data, as other methods such as LSTM seem to better deal with them [63].

Regarding LSTM, it is to be said that it is a type of RNN showing great performance when it comes to predicting time series data [64]. However, recent studies suggest that the combination of CNN in one dimension and LSTM help extract sequential features of time series [65], which allows to recognize local patterns in a sequence [66], thus leading to the achievement of more accurate results related to predictions, as well as precision and stability [67].

With respect to the design and implementation of an estimator based on CNN-LSTM, it is going to be outlined just at a high level for simplicity purposes [68]. Regarding the CNN, Figure 20 shows the composition of a neural network, where the input vector is represented by x_1 and x_2, whilst w_1 and w_2 account for the weights to be applied to each component of the input vector, whereas *b* states for bias. Then, simple linear weighting is performed to achieve *z*, and in turn, a nonlinear activation function so as to perform non linear transformation, which leads to the output.

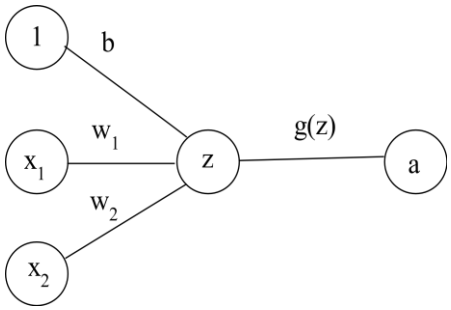

Figure 20. Composition of a Neural Network

With regards to LSTM, Figure 21 depicts the working principle of it, where σ represents the activation function sigmoid, tanh is the activation function related to hyperbolic tangent, $\times$ is the pointwise multiplication and $+$ is the pointwise addition. It is to be said that LSTM enhances the accuracy of traditional RNN, as it solves their gradient dispersion problem and their long-range dependency issue [69].

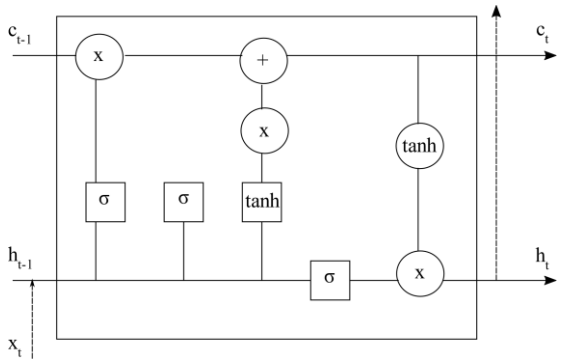

Figure 21. Working paradigm of LSTM

6. FORMAL ALGEBRAIC SPECIFICATION OF THE MODEL

As exposed above, all five scenarios included in the framework proposed depend on *k* and *n*, thus a formal algebraic specification of a scenario using generic values of those variables may apply to any of those scenarios. In order to compose such a generic layout, an abstract process algebra called Algebra of Communicating Processes (ACP) is going to be used, as it permits to just focus on the relationships occurring among the objects taking part on the scenario, leaving aside the real nature of each of these objects [70]. Hence, the use of ACP allows to attain a formal algebraic specification of the model in a pretty simple manner by mathematically describing how the nodes being part of the model interact to each other. In other words, ACP is going to be used to achieve a formal algebraic specification of a toroidal *k*-ary grid topology with generic values, which may account for any of the five instances described in sections 3 and 4, and which have been described in section 5 as the components of the model proposed.

In this context, it may be convenient to clearly define some concepts used herein, such as object, entity, device and node, which are all taken as synonyms of the computing units being part of a data center, whilst port and interface refer to the end point of a link from one object to one of its directly connected peers, whereas channel, path and link do to the communication path between to neighboring object. Hence, a link gets an object connected to another object by a direct channel between them. Furthermore, focusing on a toroidal *k*-ary grid topology, every object is identified by a string of *n* symbols, each one with *k* available values ranging from 0 to *k*-1. Moreover, each of those symbols is associated with a dimension going from 1 to *n*, in a way that if objects are identified by a string of 2 symbols, the first symbol is assigned to the dimension 1, whereas the second one is associated to dimension 2. Besides, an analogous reasoning may also be applied to objects identified with a string of any number of symbols, where all objects within the same toroidal *k*-ary grid topology are identified with a string of the same amount of symbols, although each object must have a unique string.

It is to be noted that ACP has 2 atomic actions, such that sending a message and receiving it. If a generic message is represented by *d* and the channel involved is done by port *p*, then the former action is stated by $s_p(d)$, whereas the latter is done by $r_p(d)$. Furthermore, actions interact with others by means of operators, such as the sequential one ($\cdot$), where actions are executed one after the other, the alternate one (+), where just one of the actions get executed, the concurrent one (||), where actions get run in a parallel fashion, or the conditional one: $(True \lhd condition \rhd False)$, where first the condition is evaluated, and if it is met, then the actions on the left hand side are run, whilst if it is not fulfilled, then actions on the right hand side are executed [71].

Therefore, the model of a single node may easily be constructed by combining atomic actions with the aforesaid operators in order to reflect how the message exchange dynamics take place in such a node. This way, a node *i* is called V_i and the total amount of nodes is k^n, as described in previous sections, which results in identifying each node with a number going from 0 all the way to k^{n-1}.

Furthermore, each node must have *k*-1 ports per dimension because the symbol associated to each dimension has *k* possible values, those ranging from 0 to *k*-1, which means that departing from a any of those values, there are *k*-1 possible new values to move to. Therefore, considering that each node has *n* dimensions and an extra port for internal usage is also defined, then it results in each node having actually $n(k-1)+1$ ports overall towards its neighboring nodes, those going from port 0 all the way to port $n(k-1)-1$. This is depicted in Figure 22, where each single dimension involves *k*-1 ports, along with an additional one branded as internal port, whose identifier is $(k-1)n$.

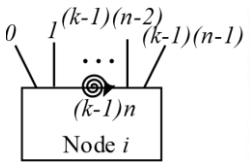

Figure 22. Port layout in a node i within a toroidal k-ary grid

Hence, those ports are labeled related to the dimension where each port is pointing to, such that the ports involved in a dimension m={1…n} will range from port $(k-1)\cdot(m-1)$ all the way to $(k-1)\cdot m - 1$. In the particular case of binary alphabets, where *k*=2, there is only one port per dimension, namely, *m*-1. Besides, port $(k-1)\cdot n$ is reserved to indicate an internal port, which allows to be the source node *a* so as to send messages towards others, or otherwise, to be the destination node *b* in order to receive messages from others. Obviously, the internal port will be *n* for the binary case.

Additionally, each atomic action has two parameters, such as the device and the port involved. On the one hand, it is to be said that the node identifier is always cited as V_i, where *i* will depend on which node is concerned, and

it is always quoted as the subindex on the left, just before the comma. On the other hand, the port identifier will be expressed as *p* at the receiving ends as the receive action at any port whatsoever is the one triggering the appropriate forwarding action towards the shortest path to destination, whereas it will be quoted as an arithmetic expression at the sending ends as such an equation will calculate the shortest way to get to destination. Anyway, the port expression is always cited as a subindex on the right, just after the comma.

Furthermore, all those elements may be put together in order to describe the behavior of a given node V_i in an algebraic fashion by means of ACP. To start with, all nodes are set in a concurrent manner as they work in a distributed way, thus *i* ranges from 0 to k^n - 1. Moreover, all ports in every single node are also set in a concurrent fashion as they are all ready to either send or receive messages, hence *p* ranges from 0 to (*k*-1)·*n*. The first action stands for receiving a message on any port of a particular node *i*, and at that point, it is checked whether that node *i* is the destination node *b* of such a message. If that is the case, the message is processed by this node, which is expressed as sending the message towards its internal port, which happens to be (*k*-1)·*n*. Otherwise, the message is forwarded on towards its neighboring nodes meeting the condition that values assigned to the same dimension differ.

At this stage, it is to be mentioned that obtaining the symbols d_m in a *k*-ary alphabet of a number in a decimal format *d* is easily done by taking the first term of (3), where *m* takes the corresponding value for each dimension, namely, from 1 all the way to *n*. Therefore, the diverse symbols representing *d* in a *k*-ary alphabet are obtained from the least significative one to the most significative one. In this equation, $\lfloor x/y \rfloor$ stands for integer division of *x* by *y*, whereas $x_{|k}$ does for *x* arithmetic modulo *k*.

$$d_m = \sum_{m=1}^{n} \left\lfloor \frac{d}{k^{m-1}} \right\rfloor_{|k} \cdot k^{m-1} \tag{3}$$

This way, the last part of the expression, located in the second line of it, just checks out symbol by symbol, each one belonging to a certain dimension *m*, on node *i*, which is the current one, and on node *b*, which is the destination one, so as to verify whether each corresponding pair of symbols match or mismatch. If the former happens, it show that symbols at a given position in both nodes are concordant, hence no action need to be taken, which is expressed as ∅. However, if the latter happens, it exposes that symbols at a particular position in both nodes are discordant, thus the path towards *b* will be found through port (4), which happens to account for port (5) in the binary case.

$$(k-1)\cdot(m-1) + \left\lfloor \frac{b}{k^{m-1}} \right\rfloor_{|k} \tag{4}$$

$$(m-1) + \left\lfloor \frac{b}{2^{m-1}} \right\rfloor_{|2} \tag{5}$$

All of those implications have been put together in (6) in order to display the model of a generic node V_i, which represents a recursive equation, meaning that it is always waiting for a new message to deal with, mirroring the behavior of a server. Furthermore, the ∨ operator of logic disjunction ($\bigvee_x^y f(d)$) represents that any of the referred items in each case, in the range from x to y, may be applied to the corresponding function $f(d)$. Specifically, the first instance of ∨ stands for all the nodes in the whole toroidal topology, the second one does for all the ports in each node and the third one does for all the ports available to forward traffic from the current node *i* to destination node *b*, as long as they meet the requirement of being equal cost multiple paths where such a cost is minimum.

$$V_i = \bigvee_{i=0}^{k^n-1} \bigvee_{p=0}^{(k-1)n} \left(r_{Vi,p}(d) \cdot \left(\bigvee_{m=1}^{n} \left(s_{Vi,(k-1)\cdot(m-1)\cdot\left\lfloor \frac{b}{k^{m-1}} \right\rfloor_{|k}} (d) \lhd \left\lfloor \frac{i}{k^{m-1}} \right\rfloor_{|k} \neq \left\lfloor \frac{b}{k^{m-1}} \right\rfloor_{|k} \rhd \emptyset \right) \right) \right) \cdot V_i \tag{6}$$

Concurrent operators are employed so as to describe distributed behavior, although such operators may be expressed in terms of left merges and communications, which allow to obtain easier mathematical expressions to work with, and in order to achieve that, the Expansion Theorem of Bergstra and Klop does the job [72]. Expression (7) depicts such a theorem, where the left merge operator is cited as (⊢), which accounts for the execution of the first term, whilst the rest of terms run concurrently afterwards, whereas the communication operator is quoted as (|), which states for the concurrent execution of the objects selected, whilst the other terms run in a concurrent fashion after that. It is to be noted that X^i stands for all objects *X* running concurrently, except X^i, whereas $X^{i,j}$ does consequently [73].

It is to be said that the amount of terms given by this expression is just the triangular number corresponding to the sum of objects running concurrently. Hence, it seems clear that such a quantity grows as the number of objects rises because all feasible combinations among them also increase. However, a specific operator may be applied to the aforementioned expression in order to consider just the relevant cases where communication happens in a particular channel, whilst bringing the rest of cases to deadlock, thus dropping them off.

$$(X_1 || \ldots || X_n) = \sum X_i \vdash X^i + \sum (X_i | X_j) \vdash X^{i,j} \tag{7}$$

This is done by the encapsulation operator, which cancels all internal atomic actions, whilst allowing communications along internal channels. That operator is specified by ∂_H, where *H* represents a set with all atomic actions. Furthermore, after applying this operator to all objects running in a concurrent fashion, the sequence of events of the model gets revealed in an algebraic manner,

thus basically presenting such events either sequentially or alternatively according to the conditions imposed [74].

Therefore, expression (8) is obtained when first applying the concurrent operator with all nodes within a generic toroidal topology, and right after, applying the encapsulation operator so as to keep all relevant terms to get the sequence of events, whilst taking out all irrelevant ones for not resulting into communication, which go deadlock. Moreover, it is to be noted that the terms appearing in expression (6) have been properly rearranged herein for clarity purposes.

Focusing on expression (8), every node is run concurrently, hence i ranges from 0 to k^n-1, whereas every port in each node is ready to go, thus p ranges from 0 all the way to (k-1)n. The sequence of events starts with a message being received in a given node i at any port p, which is followed by checking whether that node i is the destination node of the message, referred to as node b. If such an evaluation is negative, then both node identifiers are going to be compared symbol by symbol looking for all discordant pairs, and in such cases, the message will be sent over through the port at i heading for b in the corresponding dimension. After having finished all comparisons, the message will get to the destination node b, and in turn, it will be sent towards its internal port. On the other hand, if the first evaluation between i and b is positive, then the message will also be sent towards its internal port.

$$\bigvee_{i=0}^{k^n-1} \partial_H(V_i) = \bigvee_{i=0}^{k^n-1} \bigvee_{p=0}^{(k-1)n} \left(r_{Vi,p}(d) \cdot \left(\bigvee_{m=1}^{n} \left(c_{Vi,(k-1)\cdot(m-1)\cdot\left\lfloor \frac{b}{k^{m-1}} \right\rfloor_{|k}}(d) \lhd \left\lfloor \frac{i}{k^{m-1}} \right\rfloor_{|k} \neq \left\lfloor \frac{b}{k^{m-1}} \right\rfloor_{|k} \rhd \emptyset \right) \right) s_{Vi,(k-1)n}(d) \right) \cdot \partial_H(V_i) \right) \lhd i \neq b \rhd \emptyset \quad (8)$$

7. VERIFICATION OF THE MODEL

It is to be noted that ACP is a process algebra without the concept of time, hence its target is not about performance regarding time, but it is about getting formal algebraic specifications related to systems composed by a number of objects working in a concurrent manner, which may be further verified in an algebraic fashion.

In order to verify the ACP specification of a model shown above, it is necessary to apply the abstraction operator in order to mask internal communications, which leads to unveil the external behavior of the model. The abstraction operator is stated as τ_I, where I represents a set with all internal communications, such that after its application, only external actions will be present in the model, as all internal communications get masked [75].

$$\bigvee_{i=0}^{k^n-1} \tau_I\left(\partial_H(V_i)\right)_i = \bigvee_{i=0}^{k^n-1} \bigvee_{p=0}^{(k-1)n} \left(r_{Vi,p}(d) \cdot s_{Vi,(k-1)n}(d) \right) \cdot \tau_I\left(\partial_H(V_i)\right) \quad (9)$$

On the other hand, it is necessary to obtain the external behavior of the real system by means of ACP notation. In other words, the external behaviour of the model is composed by a message getting into the system and another message getting out of it after being processed, whilst the message exchange being performed inside the system is irrelevant, thus considering the system as a black box.

This is expressed in equation (10), where the recursive variable X stands for the external behavior of the real system. Basically, this expression represents a node i receiving a message, followed by another node i sending it, which accounts for the behavior of X viewed from outside the system. Hence, the external behavior stands for the messages getting into the system and those getting out of the system, thus ignoring any type of internal transaction.

$$X = \bigvee_{i=0}^{k^n-1} \left(r_{Vi,p}(d) \cdot s_{Vi,(k-1)n}(d) \right) \cdot X \quad (10)$$

At this point where the external behavior of both the model and the real system have been obtained by means of ACP notation, it is feasible to compare if both share the same string of actions and the same branching structure. If this is the case, it may be said that the model and the real system are rooted branching bisimilar, which is a sufficient condition to get a model verified [76].

In this case, it is easily viewed that expressions (9) and (10) are both recursive equations being multiplied by the same factors in the same order. Moreover, the branching structure of both is linear, as only sequential operators multiplying the same factors are involved in both cases. Therefore, both expressions are clearly rooted branching bisimilar [77], hence, it may be stated that the model has been verified, as exposed in (11). In other words, it has been proved that both expressions have the same external behavior [78].

$$\bigvee_{i=0}^{k^n-1} \tau_I\left(\partial_H(V_i)\right) \Leftrightarrow X \quad (11)$$

8. CONCLUSIONS

In this paper, a dynamic framework based on toroidal k-grids have been proposed as a manner to achieve good performance whilst saving energy. First of all, toroidal topologies have been introduced, such as de Bruijn tori and k-ary n-cubes, which are other sorts of toroidal topologies with diverse characteristics. However, toroidal k-ary grids do have the specific feature of having all nodes with just one discordant symbol at just one hop away.

Five toroidal k-ary grid topologies have been proposed, all of those with a small number of nodes, which makes

the overall structure more manageable. It is to be said that the target of those topologies is to use them in edge computing environments, where the number of users is limited, and so are the necessary servers to deal with them.

Among the topologies revised, one of them is the most basic one, as it only has 4 nodes, where k=2 and n=2. As this is the smallest one proposed, it is considered as the case base scenario because the minimal number of nodes up implies that the rest of the nodes will be idle, thus energy saving rate will get higher, leading to lower the carbon footprint of the whole system.

However, as traffic throughput will increase at any time, two lines of growth have been defined, such as n-line and k-line. Regarding the former, it is composed by two topologies, where the first one stands for k=2 and n=3, whilst the last one does for k=2 and n=4. This case is bound for higher levels of traffic coming in, such that the first one involves 8 nodes and the last one does 16 nodes.

With respect to the latter, it is formed by other two topologies, with the first one having k=3 and n=2, whereas the last one doing k=3 and n=3. That case is addressed for much higher levels of traffic getting in, where the first one involves 9 nodes and the last one does 27 nodes. By looking at the first step in both cases, there is just one node of difference, hence it does not really make a difference which one to choose. Nonetheless, the second step in both cases does make a difference, because the n-line offers 16 nodes, whereas the k-line does 27 nodes.

Hence, it is crucial to work with external parameters in order to make good predictions about the traffic coming in the near future. Those parameters are network traffic throughput, as well as historical data, recent baseline measurements and network intelligence information. Therefore, in order to extract the best values for parameters k and n at any given time, the design of a CNN-LSTM system has been just outlined in order to perform such calculations, although it should be properly trained to find out the appropriate settings in order to extract the most convenient values of k and n according to the current and upcoming external circumstances related to network traffic.

Regarding the future directions of this work, the point would be to put this design into practice by first collecting a wide dataset regarding the aforesaid parameters, such as network traffic throughput, historical data, recent baseline measurements and network intelligence information. Then, such a dataset will be split into three subsets, where one holds 60% of the total, whilst the remaining part is equally distributed between the other two, thus having 20% each.

On the one hand, the first subset will be devoted to training a classification model for the CNN-LSTM system in order to obtain the most convenient values for k and n for a given instance of the aforementioned parameters. On the other hand, the second subset will be dedicated to cross validating the model obtained, whereas the last subset will be used to testing the model achieved in order to check its F1 score, which is a machine learning evaluation metric related to the model's accuracy, which actually depends on both its precision and its recall.

Additionally, a formal algebraic model has been specified for a generic toroidal k-ary grid by abstracting away the values of k and n. In fact, this model has been designed for any given value of k and n, hence it actually may be applied to all of the five cases presented.

Eventually, the model obtained has also been verified by proving that the external behavior of the model proposed and the real system share both the same string of actions and the same branching structure.

ACKNOWLEDGEMENT

I would like to thank Orhan Aslan, who helped me with his valuable assistance and contributions in the creation of this study.

DECLARATION OF ETHICAL STANDARDS

The author (s) of this article declare that the materials and methods they use in their studies do not require ethics committee permission and / or legal-specific permission.

AUTHORS' CONTRIBUTIONS

Pedro Juan ROİG: Wrote the manuscript, performed the calculations and analyzed the results.
Salvador ALCARAZ: Analyzed the results.
Katja GILLY: Analyzed the results.
Cristina BERNAD: Analyzed the results.
Carlos JUIZ: Analyzed the results.

CONFLICT OF INTEREST

There is no conflict of interest in this study.

REFERENCES

[1] Agarwal G.K., Magnusson M. and Johanson A., "Edge AI Driven Technology Advancements Paving Way Towards New Capabilities", ***International Journal of Innovation and Technology Management,*** 18(1): 2040005, (2021).

[2] Fragkos G., Lebien S. and Tsiropoulou E.E., "Artificial Intelligent Multi-Access Edge Computing Servers Management", ***IEEE Access***, 8: 171292–171304, (2020).

[3] Bi S. et al., "A Survey on Artificial Intelligence Aided Internet-of-Things Technologies in Emerging Smart Libraries", ***Sensors***, 22(8): 2991, (2022).

[4] Alshamrani M., "IoT and artificial intelligence implementations for remote healthcare monitoring systems: A survey", ***Journal of King Saud University Computer and Information Sciences***, 34(8A): 4687–4701, (2022).

[5] Rego A., González-Ramírez P.L., Jiménez J.M. and Lloret J., Artificial intelligent system for multimedia

services in smart home environments, ***Cluster Computing***, 25: 2085–2105, (2021).

[6] Lv Z., "Practical Application of Internet of Things in the Creation of Intelligent Services and Environments", ***Frontiers in the Internet of Things***, 1: 912388, (2022).

[7] Said S. et al., "AIOT-Arch: Furthering Artificial Intelligence in Big Data IoT Applications", ***Materials Science and Engineering***, 1051: 012008, (2021).

[8] Kebande V.R., "Industrial internet of things (IIoT) forensics: The forgotten concept in the race towards industry 4.0", ***Forensic Science International: Reports***, 5: 100257, (2022).

[9] Har L.L. et al., "Revolution of Retail Industry: From Perspective of Retail 1.0 to 4.0", ***Procedia Computer Science***, 200: 1615–1625, (2022).

[10] Bzai J. et al., "Machine Learning-Enabled Internet of Things (IoT): Data", ***Applications, and Industry Perspective. Electronics***, 11: 2676, (2022).

[11] Mansour R.F., "Artificial intelligence based optimization with deep learning model for blockchain enabled intrusion detection in CPS environment", ***Scientific Reports***, 12: 12937, (2022).

[12] Nagajayanthi B., "Decades of Internet of Things Towards Twenty-first Century: A Research-Based Introspective", ***Wireless Personal Communications***, 133: 3661–3697, (2022).

[13] Kraus et al., "Digital Transformation: An Overview of the Current State of the Art of Research", ***SAGE Open***, 11(3): 1–15, (2021).

[14] Wang Z. et al., "A Survey on Recent Advanced Research of CPS Security", ***Applied Sciences***, 11(3): 3751, (2021).

[15] Greer C., Burns M.J., Wollman D. and Griffor E., "***Cyber-Physical Systems and Internet of Things***", Special Publication (NIST SP) - 1900-202, National Institute of Standards and Technology, Gaithersburg, MD, USA, 1–61, (2019).

[16] Tyagi A.K, and Sreenath N. "***Handbook of Research of Internet of Things and Cyber-Physical Systems - An Integrative Approach to an Interconnected Future***", Apple Academic Press, 1st edition, New York City, NY, USA, 1–680, (2022).

[17] Bagula A., Ajayi O. and Maluleke H., "Cyber Physical Systems Dependability Using CPS-IOT Monitoring", ***Sensors***, 21(8): 2761, (2021).

[18] Nandhini R.S. and Lakshmanan R.A., "Review of the Integration of Cyber-Physical System and Internet of Things", ***International Journal of Advanced Computer Science and Applications (IJASA)***, 13(4): 459–465, (2022).

[19] Tao F., Zhang M. and Nee A., "Digital Twin, Cyber–Physical System, and Internet of Things", in "***Digital Twin Driven Smart Manufacturing***", chapter 12, Academic Press, 1st edition, Cambridge, MA, USA, 243–259, (2019).

[20] Singh K.D. and Sood S.K., "5G ready optical fog-assisted cyber-physical system for IoT applications", ***IET Cyber-Physical Systems: Theory and Applications***, 5(2): 137–144, (2020).

[21] Park K.J., Kang K., Wang Q. and Lee D., "Real-time Internet of things and cyber-physical systems", ***Transactions on Emerging Telecommunication Technologies***, 30(4): e3466, (2019).

[22] Chen X. et al., "IoT cloud platform for information processing in smart city", ***Computational Intelligence***, 37(3): 1428–1444, (2021).

[23] Moura P., Moreno J.I., López-López G. and Álvarez-Campana M., "IoT Platform for Energy Sustainability in University Campuses", ***Sensors***, 21(2): 0357, (2021).

[24] Humayun M., "Role of Emerging IoT Big Data and Cloud Computing for Real Time Application", ***International Journal of Advanced Computer Science and Applications (IJASA)***, 11(4): 494–506, (2020).

[25] Mörth O., Emmanoulidis C., Hafner N. and Schadler M., "Cyber-physical systems for performance monitoring in production intralogistics", ***Computer & Industrial Engineering***, 142: 106333, (2020).

[26] Sabireen H. and Neelanarayanan V., "A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges", ***ICT Express***, 7(2): 162–176, (2021).

[27] Hamdan S., Ayyash M. and Almajali M., "Edge-Computing Architectures for Internet of Things Applications: A Survey", ***Sensors***, 20(22): 6441, (2020).

[28] Varsha R., Nair S.M. and Tyagi A.K., "The Fog/Edge Computing: Challenges, Serious Concerns, and the Road Ahead", in "***Advanced Analytics and Deep Learning Models"***, chapter 16, Scrivener Publishing LLC, Beverly, MA, USA, 365–389, (2022).

[29] Chegini H., Naha R.K., Mahanti A. and Thulasiraman P., "Process Automation in an IoT–Fog–Cloud Ecosystem: A Survey and Taxonomy", ***IoT***, 2(1): 92–118, (2021).

[30] Román R., López J. and Mambo M., "Mobile Edge Computing, Fog et al.: A Survey and Analysis of Security - Threats and Challenges", ***Future Generation Computer Systems***, 78: 680–698, (2018).

[31] Xiaoyi Z. et al., "IoT driven framework based efficient green energy management in smart cities using multi-objective distributed dispatching algorithm", ***Environmental Impact Assessment Review***, 88: 106567, (2021).

[32] Restrepo L., Aguilar J., Toro M. and Suescún E., "A sustainable-development approach for self-adaptive cyber–physical system's life cycle: A systematic mapping study", ***Journal of Systems and Software***, 180:, 111010, (2021).

[33] Du W. et al., "Fault-Tolerating Edge Computing with Server Redundancy Based on a Variant of Group Degree Centrality", in: "***Lecture Notes in Computer Science"***, 12571, Springer, Cham, 198–214, (2020).

[34] Liu Y. et al., "A Novel Load Balancing and Low Response Delay Framework for Edge-Cloud Network Based on SDN", ***IEEE Internet of Things Journal***, 7(7): 5922–5933, (2020).

[35] Pereira F. et al., "N.B. Challenges in Resource-Constrained IoT Devices: Energy and Communication as Critical Success Factors for Future IoT Deployment", ***Sensors***, 20(22): 6420, (2020).

[36] Marsh-Hunn D. et al., "A Comparative Study in the Standardization of IoT Devices Using Geospatial Web Standards", ***IEEE Sensors Journal***, 21(4): 5512–5528, (2021).

[37] Laroui M. et al., "Edge and fog computing for IoT: A survey on current research activities & future directions", ***Computer Communications***, 180(C): 210–231, (2021).

[38] Nizetic S., Solic P., López-de-Ipiña González-de-Artaza D. and Patrono L., "Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future", ***Journal of Cleaner Production***, 274: 122877, (2020).

[39] Lawal K. and Rafsanjani N. "Trends, benefits, risks, and challenges of IoT implementation in residential and commercial buildings", ***Energy and Built Environment***, 3(3): 251–266, (2022).

[40] Chen S., Jiao L., Liu F. and Wang L. "EdgeDR: An Online Mechanism Design for Demand Response in Edge Clouds", ***IEEE Transactions on Parallel and Distributed Systems***, 33(2): 343–358, (2022).

[41] Huang C. and Shen S.H., "Enabling Service Cache in Edge Clouds", ***ACM Transactions on Internet of Things***, 2(3): 18, (2021).

[42] Wang H. et al., "Error-Compensated Sparsification for Communication-Efficient Decentralized Training in Edge Environment", ***IEEE Transactions on Parallel and Distributed Systems***, 33(1): 14–25, (2022).

[43] Girolami M. et al., "A mobility-based deployment strategy for edge data centers", ***Journal of Parallel and Distributed Computing***, 164: 133–141, (2022).

[44] Maganelli M., Soldati A., Martirano L. and Ramakrishna S. "Strategies for Improving the Sustainability of Data Centers via Energy Mix, Energy Conservation, and Circular Energy", ***Sustainability***, 13(11): 6114, (2022).

[45] Abreha H.G., Hayajneh M. and Serhani M.A., "Federated Learning in Edge Computing: A Systematic Survey", ***Sensors***, 22(2): 450, (2022).

[46] Liu J. et al., "Exploring Query Processing on CPU-GPU Integrated Edge Device", ***IEEE Transactions on Parallel and Distributed Systems***, 33(12): 4057–4070, (2022).

[47] Roig P.J., Alcaraz S., Gilly K., Bernad C. and Juiz C., "An efficient architecture for edge data center networks", in ***Proceedings of 14th ICT Innovations Conference***, 29 September - 1 October 2022, Skopje, North Macedonia, 131–146, (2022).

[48] Bhattacharya T. et al., "Capping carbon emission from green data centers", ***International Journal of Energy and Environmental Engineering***, Springer, (2022).

[49] Wang S., Yu Y., Jiang T. and Nie J. "Analysis on carbon emissions efficiency differences and optimization evolution of China's industrial system: An input-output analysis", ***Plos One***, 0258147, (2022).

[50] Wang J., Li J. and Zhang Q., "Does carbon efficiency improve financial performance? Evidence from Chinese firms", ***Energy Economics***, 104: 105658, (2021).

[51] Kapinya J.B., "Evolutionary Computing Solutions for the de Bruijn Torus Problem", ***Master's Thesis***, Vrije Universiteit, Amsterdam, The Netherlands, (2004).

[52] Shih Y.K. and Kao S.S., "One-to-one disjoint path covers on k-ary n-cubes", ***Theoretical Computer Science***, 412: 4513–4530, (2011).

[53] Benelli G. et al., "Data Science and Machine Learning in Education", ***arXiv***, arXiv:2207.09060, (2022).

[54] Karaahmetoglu E., Ersöz S., Türker A.K., Ates V. and Inal A.F., "Evaluation of Profession Predictions for Today and the Future with Machine Learning Methods: Emperical Evidence From Turkey", ***Journal of Polytechnic***, 26(1): 107–124, (2023).

[55] Alzubaidi L. et al., "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions", ***Journal of Big Data***, 8: 53, (2021).

[56] Fang W., Love P.E., Luo H. and Ding L., "Computer vision for behaviour-based safety in construction: a review and future directions", ***Advanced Engineering Informatics***, 043: 100980, (2020).

[57] Ben Atitallah S., Dris M., Boulila W. and Ben Ghézala H., "Randomly initialized convolutional neural network for the recognition of COVID-19 using X-ray images", ***International Journal on Imaging System Technology***, 32: 55–73, (2022).

[58] Kiranyaz S. et al., "1D convolutional neural networks and applications: A survey", ***Mechanical Systems and Signal Processing***, 151: 107398, (2021).

[59] Darici M.B., "Performance Analysis of Combination of CNN-based Models with Adaboost Algorithm to Diagnose Covid-19 Disease", ***Journal of Polytechnic***, 26(1): 179–190, (2023).

[60] Farishandi A.A.K., Betancourt O. and Mamivand M., "Deep learning approach for chemistry and processing history prediction from materials microstructure", ***Scientific Reports***, 12: 4552, (2022).

[61] Roig P.J., Alcaraz S., Gilly K. and Juiz C., "Arithmetic study about energy save in switches for some data centre topologies", ***Journal of Polytechnic***, 25(2): 785–797.

[62] Cidrás-Sendra M., "High Frequency Trading via Convolutional Neural Networks", ***Bachelor's Thesis***, Universitat Politècnica de Catalunya, Barcelona, Spain, (2020).

[63] Ferreira M.D., Correa D.C., Nonato L.G. and de Mello R.F., "Designing architectures of convolutional neural networks to solve practical problems", ***Expert Systems with Applications***, 94, 205–217, (2018).

[64] Wu J.M.T. et al., "A graph-based CNN-LSTM stock price prediction algorithm with leading indicators", ***Multimedia Systems***, 00758, (2021).

[65] Gizzini A.K. et al., "CNN aided Weighted Interpolation for Channel Estimation in Vehicular Communications", ***IEEE Transactions on Vehicular Technology***, 3120267, (2021).

[66] Du S.S. et al., "How Many Samples are Needed to Estimate a Convolutional Neural Network", in: "***Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018)"***, 3-8 December 2018, Montréal, Canada, (2018).

[67] Agga A. et al., "CNN-LSTM: An efficient hybrid deep learning architecture for predicting short-term photovoltaic power production", ***Electric Power System Research***, 208: 107908, (2022).

[68] Wang M., "Prediction of the Technology Company's Stock Price through the Deep Learning Method", ***Open Journal of Modelling and Simulation***, 10(4): 428–440, (2022).

[69] Saber M. and El-Kenawy E.S.M., "Design and implementation of accurate frequency estimator depend

on deep learning", ***International Journal of Engineering & Technology***, 9(2): 367–377, (2020).

[70] Lockefeer L., Williams D.M. and Fokkink W., "Specification and Verification of TCP extended with the Window Scale Option", ***Science of Computer Programming***, 118: 3–23, (2014).

[71] Fokkink W., "***Modelling Distributed Systems***", Springer-Verlag, 2nd edition, Berlin Heidelberg, Germany, (2017).

[72] Bergstra J.A. and Klop J.W., "Verification of an alternating bit protocol by means of process algebra protocol", ***Lecture Notes in Computer Science***, 215: 9–23, (1985).

[73] Bergstra J.A. and Middleburg C.A., "Process algebra with strategic interleaving", ***Theory of Computing Systems***, 63: 488–505, (2019).

[74] Groote J.F. and Mousavi M.R., "***Modeling and Analysis of Communicating Systems***", 1st edition, MIT Press, Cambridge, MA, USA, (2014).

[75] Bergstra J. and Middelburg C.A., "Using Hoare Logic in a Process Algebra Setting", ***Fundamenta Informaticae***, 179(4): 321–344, (2021).

[76] Fokkink W., "***Introduction to Process Algebra***", Springer-Verlag, 2nd edition, Berlin Heidelberg, Germany, (2007).

[77] Roig P.J., Alcaraz S., Gilly K., Bernad C. and Juiz C. "Formal Algebraic Model of an Edge Data Center with a Redundant Ring Topology", ***Network***, 3(1): 142–157, (2023).

[78] Roig P.J., "Formal Algebraic Modelling for Fog Computing Network Architecture", ***PhD Thesis***, University of the Balearic Islands, Spain, (2022).