An efficient architecture for edge data center networks

Pedro Juan Roig[1 [0000−0002−8391−8946]], Salvador Alcaraz[1 [0000−0003−3701−5583]], Katja Gilly[1 [0000−0002−8985−0639]], Cristina Bernad[1 [0000−0001−9537−415X]], and Sonja Filiposka[2 [0000−0003−0034−2855]]

[1] Miguel Hernández University (Elche), Spain
{proig,salcaraz,katya,cbernad}@umh.es
[2] Ss. Cyril and Methodius University (Skopje), North Macedonia
sonja.filiposka@finki.ukim.mk

Abstract. Edge AI environments are ever increasing as the amount of IoT-connected devices grow, thus rising the level of carbon emissions of such ecosystems. Therefore, sustainable AI cloud/edge systems are a must in order to minimize all those emissions, where AI plays its part in gaining efficiency. In this paper, a scheme to easily organize and optimize the computing resources in an edge data center is going to be proposed, based on a specific toroidal grid topology which minimizes distance between any pair of hosts being part of it, thus reducing energy needs. That architecture may be dynamically adjusted according to the current traffic conditions and their expected variations so as to further save power consumption by using just the necessary computing assets.

Keywords: data center design · edge AI · internet of things · resource migration · toroidal topology.

1 Introduction

Edge AI may be seen as a new paradigm when dealing with IoT deployments in order to optimize performance [1]. However, the widespread deployments of edge AI domains may also produce a hype in carbon emissions, which may negatively impact the environment [2]. In fact, there are estimations about the absolute carbon footprint related to it exceeding 1000 MtCO2-eq/year [3], which might need to be tackled so as to lower it as much as possible.

In this context, the design of data centers need to take that into consideration in order to reduce the amount of carbon emissions [4]. In this sense, it has been reported that 2% of the global carbon emissions are generated by the ICT sector [5], with data centers being estimated to have the fastest growing carbon footprint within such a sector. It is to be noted that data centers efficiency is ever improving, thus making average rates of Power Usage Effectiveness (PUE) or Water Usage Effectiveness (WUE) decline by the year, although the sheer amount of new deployments makes the overall carbon emissions to considerably rise every year [6].

It is to be noted that the greatest percentage of costs in data centers are related to cooling infrastructures and energy consumption, and such a percentage keeps growing year after year, whereas the percentage of costs regarding IT equipment keeps falling, that being hardware, software and network infrastructure. Those are the current trends regarding incurred costs in data centers despite the continuous advances in virtualization, cooling and power supply [7].

Still, there are some rules out of common sense in order to make computing more environmentally sustainable [8], such as trying to extend the useful life of devices, choosing the hardware and software carefully, locating remote servers where the carbon footprint of producing energy is low, where all those initiatives may be combined by offsetting your carbon footprint in order to compensate for the necessary carbon emissions [9].

Focusing on data center network architectures, they used to be built up by a backbone of physical switches and a bunch of physical hosts playing the role of servers. However, with the rise of virtualized services several years ago, each physical server runs an hypervisor whose role is to manage a bunch of virtual servers, whereas virtual switches may play the role of physical switches [10]. In any way, traffic flows within a data center are forwarded from a source host to a destination host through the switching infrastructure, either physical or virtual, although virtualized data centers contribute to reduce green-house gas (GHG) emissions as less hardware equipment is needed [11].

In this paper, an interesting strategy to organize and optimize in a simple manner the computing resources in an edge data center is proposed. This topology makes use of the properties of toroidal grids and could be adjusted in a dynamic fashion. Hence, all necessary hosts at a given time are all connected to a bunch of active switches, thus leaving the rest of them in an idle state, thus saving power consumption and lowering the carbon footprint.

The efficiency cited in the title may be seen as twofold. First of all, it is referred to the minimal distance between any pair of nodes within the toroidal grid proposed. This is achieved by building up a specific grid topology where the node identifiers of any pair of neighbors differ in just one symbol.

That way, it is possible to travel from any source node to any destination node within the topology by taking just as many hops as the number of discordant symbols between both nodes, allowing for redundant paths when it is greater than 1. The feature of having all pair of nodes at a minimal distance does not occur in other toroidal topologies, such as de Bruijn tori or k-ary n-cubes ($k \geq 4$).

On the other hand, efficiency is also related to the fact that this design offers up to five different layouts, each of them with a specific number of nodes activated, whereas the rest of them remain in idle state, thus allowing for the establishment of energy-saving policies so as to minimize the waste of energy.

Therefore, the main goal of the paper is to achieve optimization due to taking advantage of both types of efficiency described, this is, by reducing network traffic due to the shorter distance among nodes, and also by reducing energy consumption thanks to setting unnecessary nodes in idle state, which will further impact on a reduction of carbon emissions.

The rest of the paper is organized as follows: first, section 2 presents the basics of the topology model proposed, in turn, section 3 exposes how the model works, afterwards, section 4 outlines the paths for resource migration to dynamically adapt the topology, and eventually, section 5 draws the final conclusions.

2 Basics of the topology proposed

The data center architecture proposed is made of a backbone of virtual switches whose interconnection has the shape of a toroidal topology [12], where virtual hosts are connected to those virtual switches. This way, virtual devices may go to idle state whenever they are not being used, thus saving energy and reducing the carbon footprint as much as possible, whilst being ready to go active again whenever they are needed.

The shape of toroidal topologies is established by 2 variables, where k states the number of values available in the alphabet, whilst n has a different meaning depending on the type of design selected [13]. Furthermore, the overall number of nodes within a toroidal topology is k^n, where both variables happen to be natural numbers. As an example, Figure 1 depicts a toroidal topology with 2 dimensions and 4 nodes per dimension, accounting for a total of $2^4 = 16$ nodes.

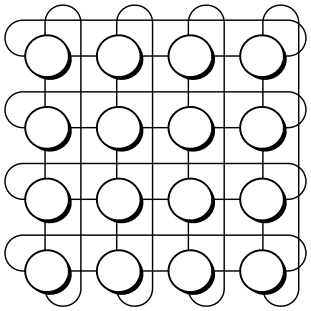

Fig. 1. Toroidal topology with dimensions 4×4.

On the one hand, toroidal topologies are employed in de Bruijn sequences with a k-alphabet and unique patterns of n-length [14], as well as their extension to higher dimensions, where a given shape portrays all available patterns of a determined size exactly once. Each node within a de Bruijn sequence gets identified by a string of n symbols from the k-alphabet, whilst it is done by a single symbol for de Bruijn tori and hypertori, where the association with the symbols of neighboring nodes portrays those different patterns [15]. This point may be appreciated in Figure 2, where the mapping of a de Bruijn torus is depicted, which happens to be the smallest possible one, namely $(4, 4; 2, 2)_2$.

On the other hand, toroidal topologies are also used in k-ary n-cubes, where nodes are identified with a string of n symbols, one for each dimension, taken

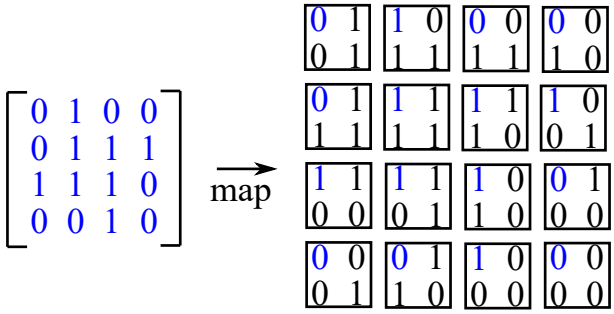

Fig. 2. Nodes and mapping of a square de Bruijn torus $(4, 4; 2, 2)_2$.

from a k-alphabet, where each of those may have any of the k values available in the alphabet considered [16]. One of the nodes located in a corner is considered to be the origin, such that all its digits are set to 0, and then, each symbol belonging to the nodes along the same dimension, going from 0 to $n - 1$, is labeled in a sequential order, from 0 up to $k - 1$, which is done for every single dimension [17], as it may be observed in Figure 3.

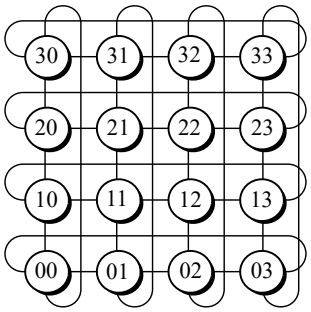

Fig. 3. Nodes of a k-ary n-cube where $k = 4$ and $n = 2$ (4-ary 2-cube).

However, an additional toroidal topology is going to be studied herein, sticking to the requirement that all pair of nodes with only one discordant symbol must be neighboring nodes, which is not the case for k-ary n-cubes where $k \geq 4$. Such a layout is going to be called binary grid in case $k = 2$, where symbols are indeed bits, as their values are either 0 or 1, or otherwise, k-ary grid if $k > 2$.

In both cases, n indicates the length of the node identifiers, as well as the number of dimensions being involved. In spite of that, those designs may also be expressed in the form of bidimensional layouts, where each node has degree n, thus making easier its representation and visualization, as well as pattern spotting. Furthermore, the path between any pair of nodes within this toroidal grid may get minimized by following the discordant symbols between them.

2.1 Binary grid

Sticking to the binary case ($k = 2$), it is to be said that the designs obtained are just n-hypercubes, as it may be shown in Figure 4, where one particular node is considered to be the origin. That node has all its digits set to zero, and in turn, a given digit of any other node are set to 0 if the value in that dimension, referred to the bit position, is the same as the origin, or otherwise, it is set to 1 if such a value is different.

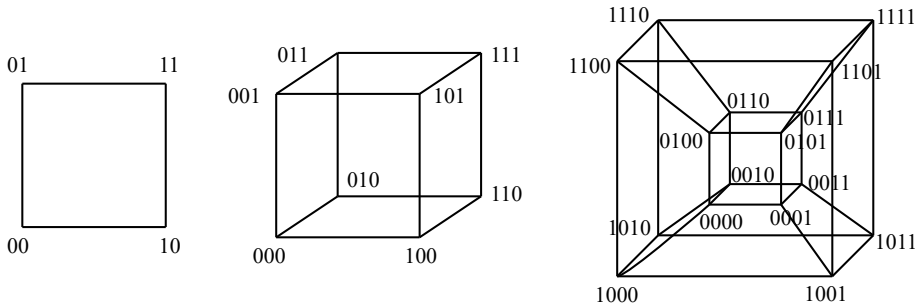

Fig. 4. Node identifiers for n-hypercube when $n = \{2, 3, 4\}$.

Taking the cases where n is even, all nodes may be redesigned as a planar square binary grid, which keeps the toroidal topology, where all elements are arranged to meet the condition that the difference between a particular node identifier with any of its neighboring nodes, those being located only one row away or one column apart, is just in one bit, which differs for each single neighbor. It is to be said that this new layout establishes definite patterns to be spotted along every row and column.

Those patterns along a given line covered by a row or a column may consist in fixing the value of some particular bits, those being either 0 or 1, whereas the rest of bits along that line may be seen as undetermined because they may vary its value, and that is why they are expressed as x in the patterns.

It is to be noted that the position of the fixed bits in the rows and in the columns are different, thus making possible that any particular binary number composed by n bits may be easily spotted by getting its corresponding bits out of crossing the fixed bits related to its corresponding horizontal and vertical patterns. Besides, the redundant paths between a given source node and a particular destination node may be easily viewed within the binary grid, even though they may also be obtained by means of arithmetic or logical operations.

On the contrary, when n is odd, all nodes may not be redesigned as a planar grid, even though an alternative bidimensional diagram is achieved by sorting all values in groups according to the overall amount of zeros in each node identifier and linking together nodes with just one discordant bit, which results in a sort of grid not being square, where the toroidal shape is not maintained. However,

it is worth noting that toroidal n-dimensional grids, are indeed found if a set of n sequentially connected planar layers are used. Anyway, the redundant paths between a particular source and a given destination may simply be observed within the diagram, although they may also be achieved by applying arithmetic or logical means.

Therefore, Figure 5 depicts the planar square grid for $n = 2$, Figure 6 exposes the planar diagram for $n = 3$, whilst Figure 7 portrays a cubic grid for $n = 3$, and Figure 8 shows the planar square grid for $n = 4$. In those cases, the different bit patterns found in rows and columns for even values of n are displayed, whereas the patterns related to the amount of bits set to zero in the node identifiers for odd values of n are exhibited, along with the composition of the row, column and layer patterns in its counterpart non-planar scheme. Moreover, higher values of n will present analogous designs, depending whether n is even or odd.

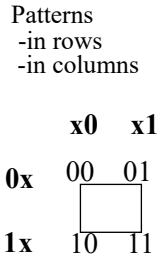

Fig. 5. Binary grid with $n = 2$ (toroidal).

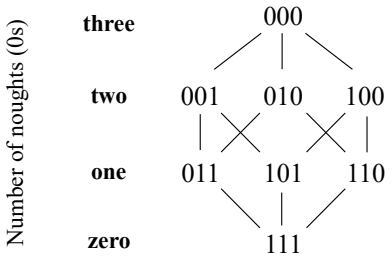

Fig. 6. Binary bidimensional diagram with $n = 3$ (non-toroidal).

2.2 *k*-ary grid

Moving on to the k-ary case, it is to be considered that the designs attained are k-ary n-cubes, as it may be shown in Figure 9, where one given node is considered as the origin, which bears all its digits set to zero, and then, moving along a

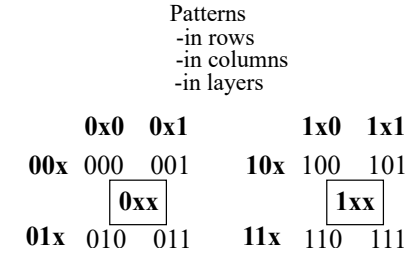

Fig. 7. Binary cubic grid with $n = 3$ (toroidal).

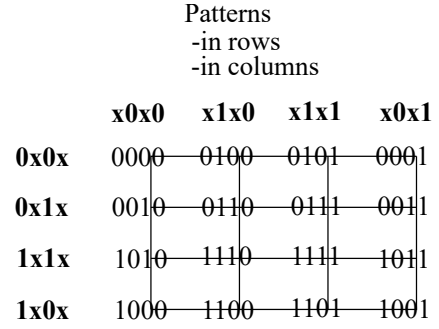

Fig. 8. Binary grid with $n = 4$ (toroidal).

certain dimension, its corresponding value increases sequentially up to reach $k-1$, whilst keeping the rest of digits unchanged along that specific dimension.

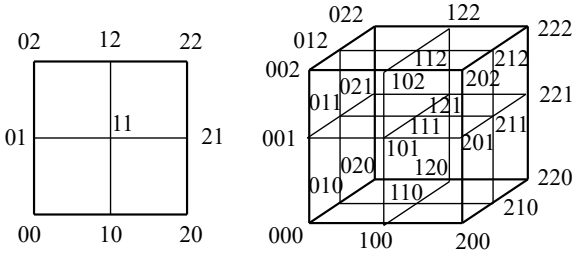

Fig. 9. Node identifiers for k-ary n-cube when $k = 3$ and $n = \{2, 3\}$.

Analogously as it happens with the binary grid case, k-ary n-cubes may also be redesigned to achieve planar k-ary grids, by replicating a similar methodology according to the parity of n, where neighbor nodes share all k-symbols but a discordant one. Hence, if n is even, then a toroidal planar square grid is obtained, resulting in definite patterns being associated to rows and columns with fixed values (ranging from 0 to $k-1$) for some digits and undetermined values for other

ones. On the other hand, if n is odd, then a kind of non toroidal planar grid is attained, with a non square shape, by sorting all values in groups regarding the total quantity of zeros in each node identifier and linking together nodes with only one discordant symbol. However, toroidal non-planar grids are still achieved when a set of n planar layers connected in a sequential manner are employed.

Therefore, Figure 10 exhibits the planar square grid for $k = 3$ and $n = 2$, whereas Figure 11 exposes the cubic grid for $k = 3$ and $n = 3$ and Figure 12 depicts the planar diagram for $k = 3$ and $n = 3$. The diverse symbol patterns located in rows and columns for even values of n are shown, whilst the patterns related to the quantity of symbols set to zero in the node identifiers for odd values of n are displayed, along with the composition of the row, column and layer patterns in its counterpart non-planar scheme. Also, the parity of n lead to analogous designs for its higher values.

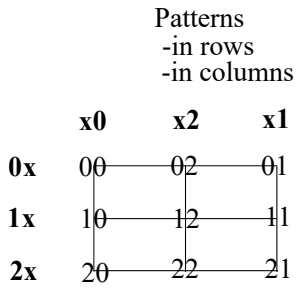

Fig. 10. k-ary grid for $k = 3$ with $n = 2$ (toroidal).

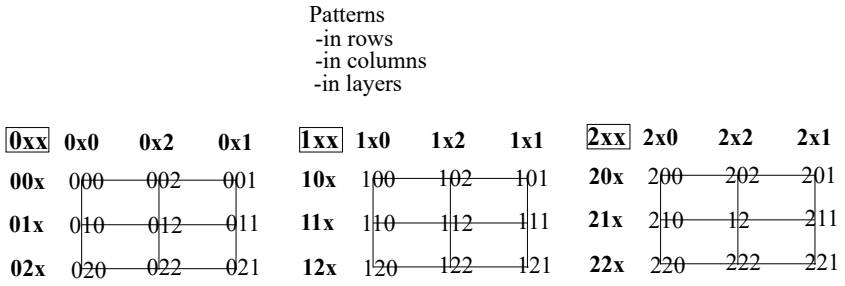

Fig. 11. K-ary cubic grid for $k = 3$ with $n = 3$ (toroidal).

3 How the model works

After having presented the basics of the topology proposed, related to undirected graphs being regular, connected, simple and Hamiltonian, where different

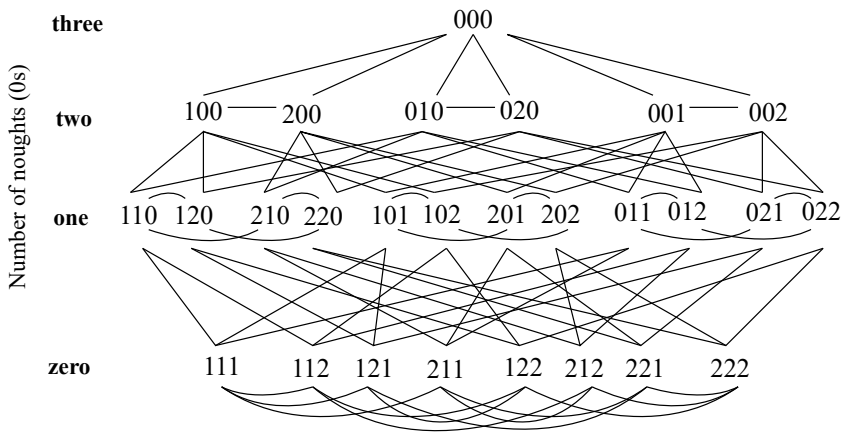

Fig. 12. K-ary bidimensional diagram for $k = 3$ with $n = 3$ (non-toroidal).

examples have been exposed, it is time to explain its behavior. First of all, it is to be considered that this paper is aimed at presenting an efficient architecture in data center networks for edge systems, hence the number of virtual switches making for the backbone is going to be small to medium size. Hence, the instances shown above may be sufficient to deal with the needs imposed by diverse traffic conditions within the edge ecosystem, as the number of virtual switches ranges from small to medium, but not going too high.

In this sense, Table 1 exposes the number of active virtual switches established for each of the instances proposed above, whose amount is given by k^n, as well as the degree of each virtual switch, which states the number of neighbors of every single node, considering that it is given by $2n$ when $k > 2$, or otherwise, n when $k = 2$, which may be summarized by the expression $n \cdot \lceil k/2 \rceil$ if $k = \{2, 3\}$.

Table 1. Active virtual switches and their degrees for the instances proposed.

	$k = 2$			$k = 3$	
	$n = 2$	$n = 3$	$n = 4$	$n = 2$	$n = 3$
Active	4	8	16	9	27
Degree	2	3	4	4	6

In view of those results, it may be seem clear that the most interesting solution for power saving is $k = 2$ with $n = 2$, because it keeps just 4 virtual switches on, whilst the rest of them are in idle state, thus being ideal for situations with low amounts of network traffic, leading to notably reduce carbon emissions.

On the other hand, when network traffic grows, two different strategic lines may be followed, such as increasing the value of n, which is suitable for situations with higher traffic rates, or otherwise, rising the value of k, which better suits

scenarios with much higher levels of traffic, as the number of virtual switches available in the latter overcomes that in the former. Hence, the former may be called n-line and the latter, k-line.

Specifically, in case of taking the n-line, then the case when $k = 2$ and $n = 3$ leads to get 8 virtual switches on, whilst that of $k = 2$ and $n = 4$ results in 16 virtual switches on. On the contrary, in case of choosing the k-line, then the scenario when $k = 3$ and $n = 2$ makes use of 9 virtual switches on, whereas that of $k = 3$ and $n = 3$ employs 27 virtual switches on. Those results show that the first step up in both cases lead to a similar number of virtual switches on, although the second step up in both cases really makes a difference, proving that the k-line is better fit to undertake much higher workloads.

It is to be remarked that power consumption will be lower for the n-line than for the k-line, as the number of active virtual switches is higher in the latter, thus a greater number of virtual switches in idle state will be found in the former. As a consequence, a certain type of tradeoff between resource availability and energy saving may be necessary, thus the topology proposed may take into account both criteria.

This way, the ideal topology may change according to the network traffic conditions, thus it makes necessary a dynamic layout, where virtual switches may go active or idle at discretion, so as to fit the most convenient topology instance at any given time out of those designs proposed above.

Hence, accurate network traffic forecast becomes a key player when dealing with sustainable computing in the topology proposed, as it may allow a more adjusted number of virtual switches being in idle state, thus permitting to shrink the carbon footprint due to the edge data center operations, without reaching the point of having constraint resources.

The rate of accuracy related to the forecast depends on the information available about the current network conditions, although other types of network intelligence may well be useful so as to predict future behavior, which indicates that the use of AI may help analyze all relevant information, thus leading to a more realistic projection of network traffic in the near future, which leads to predict the most convenient values of k and n.

Therefore, AI makes the perfect tool to swing from one topology to another in the right moment according to present traffic data, along with historical data, recent baseline measurements and network intelligence about other convenient factors, where all of those will be the input of AI calculations in order to find out the best values for parameters k and n so as to get the expected performance with the minimal carbon footprint.

The AI techniques being necessary to undertake the task of predicting the most convenient topology in a dynamic way must be first trained according to any of the most common strategies available, such as centralized learning (CL), federated learning (FL) or hybrid federated and centralized learning (HFCL) [18]. Once the necessary training is done, AI techniques may help analyze the situation by considering the current, past and expected conditions, as well as network intelligence information [19], where all relevant parameters get combined

in the appropriate way in order to predict the expected behavior of the network traffic conditions [20].

Regarding the ideal AI algorithms, it is to be said that ANN (Artificial Neural Networks) are able to deal with high flows of incoming data, and among them, CNN (Convolutional Neural Networks) are fitter to deal with greater amounts of data than RNN (Recurrent Neural Networks) or MLP (Multi Layer Perceptron). On the other hand, those AI algorithms may apply to as many nodes as necessary, even though the scenarios available for the toroidal grid proposed herein just have 4, 8, 9, 16 and 27 nodes, respectively.

In summary, the predictions made by AI techniques will lead to forecast the most convenient values for k and n. This way, n will increase when network traffic is predicted to get higher and k will grow when it is forecast to get much higher. On the other hand, the decreasing patterns will go the other way around.

Furthermore, it is to be noted that when the value of n increases, then all active node identifiers at that point need to be prepended an extra zero to adapt to the new context, whilst at the same time, the necessary nodes to construct the new layout swap from idle to active state are identified with their first symbol of node id not being zero. Also, virtual links among nodes are managed by means of AI so as to achieve the desired toroidal topology.

Similarly, when the value of n decreases, then only active nodes with their first symbol to zero keep in active state, whilst the rest of active nodes go to idle state, and in that case, all computing resources being located thereon must be migrated to the remaining nodes before going idle. Also, virtual links among nodes are dealt with by AI so as to obtain the expected topology.

The behavior of increases and decreases of variable k result in expanding or compressing the range of possible values of each symbol being part of a node identifier, even though their behavior is analogous to the changes of variable n.

4 Resource migration to dinamically adapt the topology

The use of a dynamic topology may make necessary to apply resource migration among virtual hosts through the active virtual switches. Such a virtual host migration process is mainly necessary because virtual switches may swap from active to idle, or the other way around, as virtual hosts need to be connected to active virtual switches at all times in order to be able to communicate with both their peers and the exterior network. Besides, virtual host migration may also be needed for managing purposes, such as load balancing virtual resources on physical hosts, or on the contrary, consolidating them.

In order to carry out such migration processes, routing or forwarding techniques are usually employed to get the proper interfaces along the way from the virtual source host to the virtual destination host, which require the fulfillment of routing tables or forwarding tables, respectively, both implying some processing time when undertaking the searching tasks through the corresponding table.

However, the specific layout of the topologies presented above allows for an alternative way to routing and forwarding processing by taking advantage of the

fact that there is just one discordant symbol between any pair of neighboring virtual switches. Hence, the distance between a given source and a particular destination is given by the number of discordant symbols between them both, accounting for bits in the binary case, which allows for easily discovering redundant paths by just making permutations among the elements belonging to the set of those discordant symbols.

It may seem clear that the time intervals employed in spotting the discordant symbols are faster than those used in looking up routing and forwarding tables. With regards to the former, it is done through either applying the logical XOR operator between the symbols defining a pair of node identifiers, quoted in k-ary format, or otherwise, applying arithmetic operations, such as integer divisions and modular arithmetic, between those node identifiers, cited in decimal format.

With respect to the latter, it is done through applying the logical AND operator between the destination address and all addresses within the table so as to spot a match, which appears to be more time consuming. Moreover, IP and MAC addresses are far larger than the nomenclature for node identifiers proposed herein (just a handful of symbols), resulting in even longer time intervals.

Regarding the evaluation of both methods, it happens that building up the former with NAND gates requires 4 of them, whilst the latter does just 2 of them, meaning that the former requires twice as much gates as the latter.

On the one hand, as the XOR operations are undertaken between a pair of nodes, the number of symbols composing those node IDs within the toroidal grid proposed is up to 3, where the scenario with more nodes employs of up to 27 of them, which may be expressed by using just 5 binary digits, as $2^5 = 32$. Hence, XOR operations in this context involve up to 5 bits in the worst case scenario.

On the contrary, the AND operations are to be carried out between either IP addresses or MAC addresses, where the former involve 32 bits and the latter, 64 bits. Hence, AND operations in this context employ either 32 or 64 bits, which is far more than double the XOR case. Moreover, finding the right entry on just the first try is the best case scenario, which is not usually the case, thus more searches need to be done, resulting in further operations.

In summary, it results that XOR operations with the node identifiers proposed herein occurs to be more efficient than AND operations through routing and forwarding tables in terms of the necessary operations, hence the latter loses out in network efficiency. Besides, a well-trained AI tool may raise energy saving efficiency by turning unnecessary active nodes into idle or the other way around.

Therefore, if i is the source, j is the destination, k is the size of the alphabet and p is the symbol position, ranging from the least significative, which is 0, to the most signficative, which is $k-1$, then the arithmetic way checks all symbol positions in both i and j by using (1) so as to first calculate each symbol out of its decimal format, and in turn, look for pairs of discordant symbols occupying the same position, which are spotted for each value of p where the inequality condition is met.

On the other hand, the logical way does it by using (2) so as to extract each symbol from its k-ary format, and then, search for pairs of discordant

symbols, which are found for positions where XOR operation yields 1, as 0 implies matching symbols.

$$\bigvee_{p=0}^{k-1} \left\{ \left\lfloor \frac{i}{k^p} \right\rfloor_{|k} \neq \left\lfloor \frac{j}{k^p} \right\rfloor_{|k} \right\} \tag{1}$$

$$\bigvee_{p=0}^{k-1} \left\{ i[p] \; XOR \; j[p] \right\} \tag{2}$$

Regarding the migration path, the list of movements through intermediate virtual switches to get from a source one to a destination one, both expressed in k-ary format, may require to confront the values assigned to the same symbol positions in both node identifiers so as to search all discordant symbols.

Obviously, if there are more than one discordant symbol, then redundant paths are available by making permutations on them, resulting in as many redundant paths as the factorial of the amount of those discordant symbols.

In summary, the system may start up with its lower number of virtual switches available, thus resulting from parameter $k = 2$ and $n = 2$, whereas AI is monitoring the network traffic and analyzing diverse pieces of information in order to predict when the traffic will rise and how much, so as to extend the available virtual switches as a first step by growing the value of n for higher traffic density, or otherwise, the value of k for even higher. Analogously, if the traffic keeps growing, an extra increase is obtained by getting n up an additional unit as a second step, thus a greater number of virtual switches are available. Obviously, if the first step consisted of growing k, then the second step will provide a higher amount of virtual switches.

Otherwise, if the traffic decreases, the value of n is taken back, thus reversing the second step, and if traffic keeps going down, then the first step is reversed, such as either n or k comes back to its initial value, thus reducing the number of active virtual switches. In case of any change of parameters n or k, resource migration may need to be undertaken accordingly so as to load balance such resources if the values increase, or to consolidate them if the values decrease. Hence, AI plays a crucial role in controlling the size of this topology and adapting to the network traffic conditions, and as a consequence, AI optimizes the tradeoff between performance and carbon emissions [21].

5 Conclusions

In this paper, an adaptive topology design for edge data centers has been proposed in order to reduce the carbon emissions by swapping virtual switches from active to idle state according to the predictions based on AI, where network traffic conditions are evaluated, as well as other parameters, such as historical data or network intelligence.

To start with, a kind of planar grid topology has been studied where any pair of nodes whose identifiers differ in just one symbol are linked together. Such a topology has been identified as a binary grid when used with binary values, or otherwise, a k-ary grid if used with values of an alphabet of size k. Five scenarios

have been proposed for being used in edge data centers, where the number of necessary virtual hosts attached to virtual switches are relatively small.

Each node within those topologies is identified by a string of length n, resulting in regular patterns of symbols in both rows and columns where n is even, achieving bidimensional toroidal designs. However, that condition is not met when n is odd, although the nodes have been sorted in groups according to the number of zeros within their identifiers, resulting in bidimensional designs not being toroidal, even though polydimensional toroidal grids may also be obtained.

In this context, five designs have been shown, where the one with k and n bearing their lower values may be seen as the basic layout, leading to its consideration as the initial one, as it engages the lower amount of virtual switches, thus being the one achieving a lower carbon footprint. On the other hand, AI techniques are continuously monitoring current network traffic and analyzing it along with other relevant information, such as historical data and network intelligence information, so as to predict when the traffic will increase in order to extend the topology when it gets necessary.

That extension may be done through growing n for higher rates of traffic, or otherwise, through rising k for even higher rates, where the more virtual switches are active, the more will increase the carbon emissions. Hence, monitoring ought to keep on so as to find out when the values of n or k may decrease again, thus lowering the carbon footprint as much as possible without affecting performance.

Eventually, an interesting method to forward traffic among those virtual switches has been exposed, taking advantage of the fact that neighboring nodes just has a discordant symbol, thus making possible to easily spot the redundant paths between a source and a destination by just searching for the positions where those discordant symbols are located, which may also account for redundant paths by making combinations with the set of discordant symbols found.

References

1. Vukobratović, M. et al.: A Survey on Computational Intelligence Applications in Distribution Network Optimization. Electronics 2021, 10(11):1247 (2021). https://doi.org/10.3390/electronics10111247
2. El-Mawla, N.A., Badawy, M., Arafat, H.: IoT for the Failure of Climate-Change Mitigation and Adaptation and IIoT as a Future Solution. World Journal of Environmental Engineering, 6(1), 7–16 (2019). https://doi.org/10.12691/wjee-6-1-2
3. Pirson, T., Bol, D.: Assessing the embodied carbon footprint of IoT edge devices with a bottom-up life-cycle approach. Journal of Cleaner Production, 322:128966 (2021). https://doi.org/10.1016/j.jclepro.2021.128966
4. Siddik, A.B., Shehabi, A., Marston, L.: The environmental footprint of data centers in the United States. Environmental Research Letters, 16(6):064017 (2021). https://iopscience.iop.org/article/10.1088/1748-9326/abfba1
5. Bertoldi, P., Avgerinou, M., Castellazzi, L.: Trends in data centre energy consumption under the European Code of Conduct for Data Centre Energy Efficiency. EUR 28874 EN, Publications Office of the European Union, Luxembourg (2017). https://doi.org/10.2760/358256

6. Liu, Y. et al.: Energy consumption and emission mitigation prediction based on data center traffic and PUE for global data centers. Global Energy Interconnection, 3(3), 272–282 (2020). https://doi.org/10.1016/j.gloei.2020.07.008
7. Guitart, J.: Toward sustainable data centers: a comprehensive energy management strategy. Computing, 99(6), 597–615 (2017). https://doi.org/10.1007/s00607-016-0501-1
8. Lannelongue, L., Grealey, J., Bateman, A., Inouye, M.: Ten simple rules to make your computing more environmentally sustainable. PLoS Comput Biol, 17(9): e1009324 (2021). https://doi.org/10.1371/journal.pcbi.1009324
9. El Geneidy, S. et al: The carbon footprint of a knowledge organization and emission scenarios for a post-COVID-19 world. Environmental Impact Assessment Review, 91:106645 (2021). https://doi.org/10.1016/j.eiar.2021.106645
10. Bari, F. et al.: Data Center Network Virtualization: A Survey. In: IEEE Communications Surveys & Tutorials, 15(2), 909–928 (2013). https://doi.org/10.1109/SURV.2012.090512.00043
11. Lee, H. et al.: COVID19 Led Virtualization: Green Data Center for Information Systems Research. Information Systems Management, 37(4), 272–276 (2020). https://doi.org/10.1080/10580530.2020.1818901
12. Kitayama, K.I. et al.: Torus-Topology Data Center Network Based on Optical Packet/Agile Circuit Switching with Intelligent Flow Management. Journal of Lightwave Technology, 33(5), 1063–1071 (2015). https://doi.org/10.1109/JLT.2015.2394384
13. Gardner, R.J. et al.: Toroidal topology of population activity in grid cells. Nature 602, 123–128 (2022). https://doi.org/10.1038/s41586-021-04268-7
14. de Bruijn, N.G.: A combinatorial problem. In: Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam, the Netherlads, Vol. 49(7), pp. 758–764 (1946). https://research.tue.nl/files/4442708/597473.pdf
15. Roig, P.J., Alcaraz, S., Gilly, K., Bernad, C., Juiz, C.: Review on de Bruijn shapes in one, two and three dimensions. Journal of Physics: Conference Series, 2090:012047 (2021). https://doi.org/10.1088/1742-6596/2090/1/012047
16. Xie, Y., Liang, J., Yin, W., Li, C.: The properties and t/s-diagnosability of k-ary n-cube networks. Journal of Supercomputing, 78(5), 7038–7057 (2022). https://doi.org/10.1007/s11227-021-04155-y
17. Roig, P.J., Alcaraz, S., Gilly, K., Bernad, C., Filiposka, S.: De Bruijn-based and k-ary n-cube-based algebraic models in fog environments. Communications in Computer and Information Science, 1521, 126–141 (2022). https://doi.org/10.1007/978-3-031-04206-5_10
18. Elbir, A.M., Papazafeiropoulos, A.K., Chatzinotas, S.: Federated Learning for Physical Layer Design. In: IEEE Communications Magazine, 59(11), 81–87 (2021). https://doi.org/0.1109/MCOM.101.2100138
19. Abreha, H.G., Hayajneh, M., Serhani, M.A.: Federated Learning in Edge Computing: A Systematic Survey. Sensors 22(2):450, 1–45 (2022). https://doi.org/10.3390/s22020450
20. Lotfi, I., El Bouhadi, A.: Artificial Intelligence Methods: Toward a New Decision Making Tool. Applied Artificial Intelligence 2021(10), 1–16 (2021). https://doi.org/10.1080/08839514.2021.1992141
21. Cowls, J., Tsamados, A., Taddeo, M., Floridi, L.: The AI gambit: leveraging artificial intelligence to combat climate change—opportunities, challenges, and recommendations. AI & Society, 2021:01294, 1–25 (2021). https://doi.org/10.1007/s00146-021-01294-x