POLİTEKNİK DERGİSİ

JOURNAL of POLYTECHNIC

ISSN: 1302-0900 (PRINT), ISSN: 2147-9429 (ONLINE)
URL: http://dergipark.org.tr/politeknik

Arithmetic study about energy save in switches for some data centre topologies

Veri merkezi topolojilerindeki anahtarlama aygıtlarında enerji tasarrufuna yönelik aritmetik çalışma

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

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

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

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

ORCID[4]: 0000-0001-6517-5395

Bu makaleye şu şekilde atıfta bulunabilirsiniz(To cite to this article): Roig P.J. et al., “Arithmetic study about energy save in switches for some data centre topologies”, ***Politeknik Dergisi***, 25(2): 785-797, (2022).

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

DOI: 10.2339/politeknik.810896

Arithmetic Study about Energy Save in Switches for some Data Centre Topologies

Highlights

- *Data Centre*
- *Fat Tree*
- *Folded N-Hypercube*
- *Leaf and Spine*
- *N-Hypercube*

Graphical Abstract

The work carried out in this article goes about studying two common tree-like Data Centre designs, as well as two graph-like designs, in order to find out the average distance measured in switches among hosts and among end switches. Furthermore, the idle rate of switches in those scenarios are calculated for some typical setups.

Topology	**Total Switches**		**Avg.dist.Hosts**		**Rounded Avg. dist.Hosts**		**Avg Idle Rate of Switches**	
	K=4	**K=8**	**K=4**	**K=8**	**K=4**	**K=8**	**K=4**	**K=8**
Fat Tree	20	80	3.81	4.68	20-4=16	80-5=75	16/20	75/80
Leaf Spine	10	40	2.69	2.92	10-3=7	40-3=37	7/10	37/40
N-Hyperc.	8	32	2.43	3.49	8-3=5	32-4=28	5/8	28/32
Folded N-Hy.	8	32	2.18	3.05	8-2=6	32-3=29	6/8	29/32

Table. Average idle rate of switches in the scenarios proposed.

Aim

This work is aimed at calculating the average distance among hosts and among end switches for some common Data Centre topologies. Furthermore, the average idle rates of switches in those scenarios are also found out, which allow for implementing energy save policies.

Design & Methodology

This study has been undertaken departing from the specifications of the topologies selected, and only using arithmetic operations.

Originality

This study has been carried out by means of applying arithmetic operations to calculate average distances and average idle rates in the Data Centre topologies proposed, setting three different scenarios for each one, where two of them have specific features and the other is a generic one.

Findings

There is no such a better topology for all scenarios, as results vary depending on the number of items involved.

Conclusion

The average idle rate in all topologies is high enough to consider the implementations of energy saving policies.

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.

Veri Merkezi Topolojilerindeki Anahtarlama Aygıtlarında Enerji Tasarrufuna Yönelik Aritmetik Çalışma

(Bu çalışma ECRES 2020 konferansında sunulmuştur. / This study was presented at ECRES 2020 conference.)

Araştırma Makalesi / Research Article

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

[1]Department of Computer Engineering, Miguel Hernández University, Elche, Spain
[2]Department of Computer Science, University of the Balearic Islands, Spain

(Geliş/Received : 15.10.2020 ; Kabul/Accepted : 16.02.2021 ; Erken Görünüm/Early View : 13.03.2021)

ÖZ

Paralel bilgi işlem şemaları giderek büyüdükçe, veri merkezi tesisleri hızla artıyor. Bu tür tesisler, farklı topolojilere göre tasarlanmıştır, ancak bunların çoğu yeterince kullanılmamaktadır ve bu durum, son kullanıcı bilgisayarları için ağ bağlantısı sağlayan anahtarlama aygıtlarının çoğunda enerjinin kötüye kullanılmasına neden olmaktadır. Bu çalışmada, enerji tasarrufu amacıyla performanslarını öğrenmek için daha tutarlı bir karşılaştırma yapabilmek için her tasarımda aynı sayıda kullanıcı bilgisayarı ile İri Ağaç (Fat Tree), Omurga ve Kanat (Spine and Leaf), N-Hiperküp ve Katlanmış N-Hiperküp gibi bazı yaygın veri merkezi topolojileri incelenmiştir. Her topolojinin kendi avantaj ve dezavantajları olduğu söylenebilir, bu da her birinin bir uygulamanın başlangıç koşullarına bağlı olarak en iyi seçim olabileceği anlamına gelir. Bununla birlikte, sabit gecikme ve seğirme için belirlenen koşullarda, İri Ağaç topolojilerinin veri merkezinin boyutu büyüdükçe daha iyi sonuçlar elde ettiği, diğer tarafta Omurga ve Kanat topolojilerinin bunun aksini yaptığı görülüyor.

Anahtar Kelimeler: Veri merkezi, iri ağaç, katlanmış N-Hiperküp, omurga ve kanat, N-Hiperküp.

Arithmetic Study about Energy Save in Switches for some Data Centre Topologies

ABSTRACT

Data Centre facilities are rapidly increasing as parallel computing schemes are ever growing. Those kinds of facilities are designed according to different topologies, but many of them are underutilized, leading to energy misuse in many of the switches providing network interconnection for the end hosts. In this paper, some common Data Centre topologies are studied, such as Fat Tree, Leaf and Spine, N-Hypercube and Folded N-Hypercube, in order to find out their performances in terms of energy saving purposes for the same number of hosts hanging on each design, hence obtaining a coherent comparison among them all. It is to be said that each topology has its own benefits and drawbacks, meaning that each one may be the best option depending on the initial conditions of an implementation. However, in the conditions established for steady latency and jitter, it seems that Fat Tree topologies get better results as the size of the Data Centre grows up, whereas Leaf and Spine does it otherwise.

Keywords: Data centre, fat tree, folded N-Hypercube, leaf and spine, N-Hypercube

1. INTRODUCTION

Multi-access Edge Computing (MEC) is going to be one of the key players in the Information Technology and Communication (ITC) field in the near future. It extends the Cloud Computing paradigm, bringing the computing assets to the edge of the network, thus resulting in better performances than Cloud Computing in a radio environment regarding latency, bandwidth and power consumption [1].

The overall infrastructure of MEC may be similar to that of Cloud, although the number of resources involved may be significantly scaled down as the number of wireless users getting in will be restricted to those getting access to the coverage area of the MEC [2], having the possibility of using the Cloud services located in the worldwide network as a backup solution [3].

This fact allows for the use of energy saving strategies when the number of users connected to the MEC infrastructure is small enough and the amount of resources involved in their interactions are located in certain number of hosts, thus leaving the rest of hosts in an idle state [4]. This condition may happen quite often

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

as MEC services may be offered as added value services by network operators [5], which users may have to pay, whereas general wireless services may be available for other users not willing to pay an extra fee for those premium services [6].

The target of this paper is to revise, in an arithmetic way, some of the most well-knows cloud topologies in order to study their characteristics so as to see which one better fulfills the requirements of MEC traffic with regards to power consumption [7]. It is to be noted that some switches taking part in the MEC architecture may well be put in an idle state as long as the hosts being used at a certain moment are not making use of such switches, thus resulting in energy saving schemes.

It is to be reminded that green computing is related to power efficiency, as well as cooling efficiency [8], being one of the driving forces in computing research nowadays. Furthermore, the efficient allocation of Virtual Machines (VMs) in Data Centers may also help reduce the carbon footprint related to energy consumption [9].

Therefore, the study of the occupancy rate found in some of the most widely used topologies in Data Centers may well be interesting from the point of view of green computing [10].

This paper is an extension of another one called Arithmetic Study on Energy Saving for some common Data Centre Topologies [11], presented at the ECRES 2020 Conference, held in Istanbul.

The organization of this paper is going to be as follows: first, in Section 2, a background study is shown, next, in Section 3, interconnection topologies are presented, then, in Section 4, case scenarios are defined, afterwards, in Section 5, calculations for average distance and energy saving are performed, and finally, in Section 6, some conclusions are drawn.

2. BACKGROUND

Data Centers provide the core infrastructure to meet the current requirements regarding computing and storage [12]. They are composed of a large number of servers, which is ever increasing, as well as an appropriate networking topology to get them interconnected in an efficient way.

Hence, an optimal design for a Data Center needs to take different parameters into account [13], such as a good tradeoff between data availability and security, as well as cost and energy efficiency, along with easiness to operability and maintainability.

Likewise, redundant paths between any pair of hosts is a must when dealing with Data Centers [14], and therefore, an orchestrator might be used so as to choose just one path or some of them, whilst leaving the rest aside, hence activating the switches involved and leaving out the others, depending on the load balancing policies put in place or the energy consumption scheme being in use in the whole system [15], resulting in less energy usage overall, accounting for both power and cooling. Needless to say that if all redundant paths are active, all necessary switches will do so accordingly [16].

It is to be noted that MEC and Fog have a similar target [17], as they both try to bring the Cloud infrastructure as close as possible to the end users in order for them to improve the Key Performance Indicators. However, they are standardized by different bodies, as MEC is done by ETSI group and Fog is done by Open Fog consortium.

Nonetheless, they may well be seen as complementary in nature, although Fog has a hierarchical architecture and MEC has a flat one [18]. Additionally, Fog is specifically designed to cater for the needs of IoT [19], whereas MEC does it for multiple services, such as AR/VR/MR or smart vehicles, although it is also being widely used in some IoT environments [20].

Regarding MEC implementations, it is to be remarked that these are geographically located deployments, hence, the number of hosts available depend on the extension of the coverage area, the number of users involved and its distribution within the area, which makes the main difference with Cloud implementations.

Additionally, it is to be reminded that MEC domains are quite often dedicated to moving IoT devices, which need to have their associated computing assets as close as possible in order to increase their performances, as their computing capabilities are quite limited. This point motivates the live migration of those associated computing assets, which are basically VMs, although docker containers or any other variation may also apply [21]. Nevertheless, live VM migration is the most referred term throughout literature.

This process is undertaken when one of those computing assets moves from a source host to a destination host, both located within the MEC domain. Additionally, other migration features might apply, such as Cloud support [22], where those computing assets may be migrated up to the Cloud, or otherwise to another available remote computing center being part of some locally situated facilities, such as a Fog or Mist domain.

According to literature, some models have been proposed to better undertake this process [23]. To start with, three kinds of migration were proposed, related to how computing assets are treated during migration. First of all, cold migration shuts them down before moving them, then, hot migration suspends its Operating System before moving them, and finally, live migration keeps them going whilst moving them. Obviously, the latter makes for the ideal situation, and that is why it is largely the most implemented technique.

Focusing on the latter, three parameters are its key performance indicators. The first one is downtime, being the time interval that computing assets are halted during migration. The second one is total migration time, being the elapsed time for the whole migration. And the third one is the quantity of dirty pages migrated, referring to the amount of data being changed while the migration is taking place, which may be forwarded again.

Taking into account that the amount of dirty pages may not be foreseen in advanced, as it depends on the activity carried out by the moving IoT device while the migration is being performed, the point is to obtain a tradeoff between downtime and total migration time. Some parameters have an influence on that, although memory transfer is the main point, and as such, it needs special attention.

Regarding memory transfer, three stages may be distinguished, such as push phase, given at the initial stages of the transfer process where the old VM on the source host is running, stop-and-copy, given when the old VM on the source host is halted and the new VM on the destination host is started, whilst the pages are copied across, and pull phase, given when the new VM on the destination host is running and pages not being copied yet are retrieved from the source VM.

In order to carry out the live VM migration process efficiently, the focus is usually put on one or two of those stages. However, the preferred approach is pre-copy migration, which contains a combination of bounded iterative push phase, along with a very short stop-and-copy phase. The iterative part is repeated until the transfer of all dirty pages is complete.

Therefore, considering the pre-copy live VM migration approach for a VM transference between a source host and a destination host, six stages may be differentiated, given by a predefined timeline. To wrap it all up, Figure 1 shows this timeline.

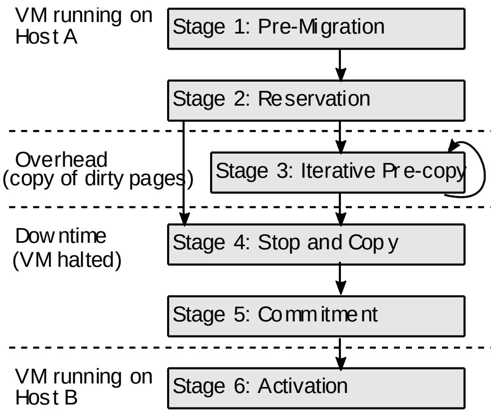

Figure 1. VM Migration timeline.

3. INTERCONNECTION TOPOLOGIES

Focusing on the topological view of parallel computing infrastructures, it has to be noted that hosts have to be interconnected by a series of switches according to a certain topology, which may impact performance, subject to the number of switches and hosts involved in each topology. Therefore, this is not the case that there is one topology better than others, but each one has its own pros and cons, depending on the implementation proposed.

Regardless of the use of those topologies, this is, no matter if they are used in an internetwork scope, such as Cloud Computing, or those used in a more restricted scope, such as Fog Computing, Mist Computing or MEC, let us consider an abstraction composed by switches being interconnected to each other according to different patterns, where those switches will be used to interconnect the different hosts where the parallel computing is taking place.

Therefore, focusing on interconnection networks, they are usually hierarchical structures divided into two different categories, such as trees and graphs. Some of the most relevant instances of the former are Fat Tree and Leaf & Spine, whereas some of the most interesting instances of the latter are N-hypercube or Folded N-hypercube.

Regarding switching topologies applied to Data Centre architectures, tree-like designs are usually preferred due to the advantages given by their modular design, whereas the graph-like designs are usually more devoted to multiprocessor implementations. Furthermore, from the point of view of maintenance, a tree-like structure makes things easier. Nonetheless, graph designs are also used in multiple network deployments due to its compactness and design adaptability.

Additionally, there are more complex topological structures regarding network interconnections for Data Centre involving tree and graph designs, as well as other types, which may achieve even better performances. Nevertheless, in this paper we are going to focus our study in the ones specified above, as being the simplest ones and the most sought after. Besides, all designs to be studied are going to be considered with an oversubscription rate 1:1, meaning that all possible links are available in the designs, as that is the optimal way to evaluate their optimal performances.

3.1. FAT TREE

It is a tree-like structure with three layers of switches, where a parameter *K* drives the whole design [24]. The lower layer is called Edge and it is the one where hosts are hanging on, the middle one is called Aggregation and the upper one is called Core. Besides, parameter *K* marks how switches in the first and second row connect to each other forming Pods, resulting in full mesh connectivity for the *K* switches evenly distributed in the different layers within a single Pod, and full mesh connectivity among Pods, which account for partial mesh connectivity between switches in the second and third row. Moreover, each switch has *K* ports, where the first and second row have half of them looking upwards and the other half looking downwards, leading to the fact that each Edge switch is holding *K/2* hosts.

In this architecture, any two hosts may be up to three hops away from each other, in a way that hosts connected to the same switch are one hop away, hosts connected to the same Pod are two hops away and hosts connected to different Pods are three hops away. If there is one hop away, the number of switches involved in a transaction will be one Edge switch [25]. On the other hand, if there are two hops away, the number of switches involved will be three, those being the Edge switches connecting both end hosts and also a single Aggregation switch connecting both Edge switches, although load balancing policies may allow more than one Aggregation switches. Eventually, if there are three hops away, the number of switches involved will be five, meaning two Edge switches, two Aggregation switches and a Core Switch linking them, although more Core switches might be involved depending on the load balancing conditions imposed [26]. A Fat Tree design for *K*=4 is presented in Figure 2.

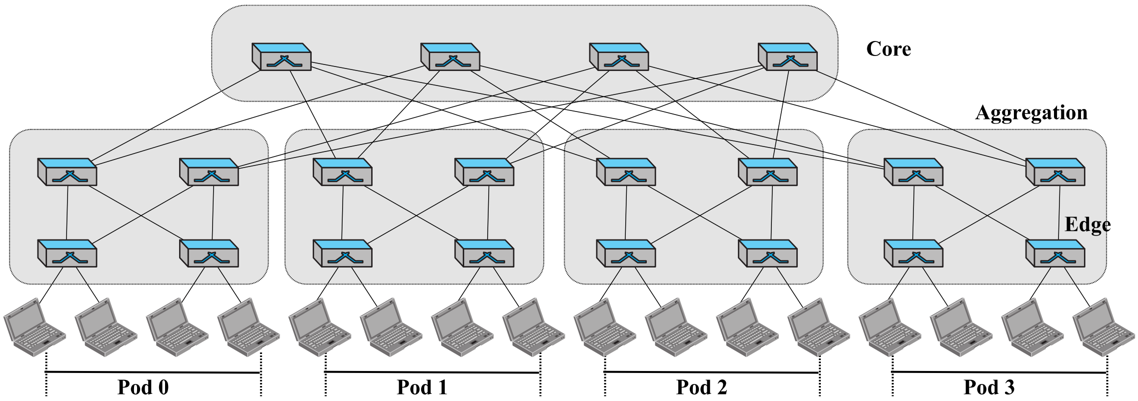

Figure 2. Fat Tree architecture for K=4 and oversubscription rate 1:1.

3.2. LEAF AND SPINE

It is also a tree-like structure with two layers of switches, where there are some degrees of freedom in the design of the topology [27], allowing for a less restrictive layout compared to Fat Tree. The lower layer is called Leaf, whereas the upper layer is called Spine. The only condition is that there must be a full mesh connectivity among all switches situated in the lower layer, the ones where host are hanging on, and all the switches situated in the upper layer.

The number of ports in each switch is not determined, as well as the number of hosts being held by each Leaf switch, and even the number of switches forming both layers, therefore, all those features are up to the designer. An instance of a Leaf and Spine implementation is shown in Figure 3, where the number of ports in all switches is 8, with Leaf switches having 4 ports looking upwards and other 4 looking at the hosts, permitting up to 4 hosts per switch [28].

This architecture allows for less distance among hosts, as any two hosts connected to the same Leaf switch are one hop away, whilst if this is not the case, they will be just two hops away. In the former case, the number of switches taking part in a single transaction is just one Leaf switch, whereas in the latter case, it will be three, such as both Leaf switches connecting the hosts involved and a single Spine switch interconnecting both, although may be more Spine switches depending on the load balancing policies put in place in the system implementation.

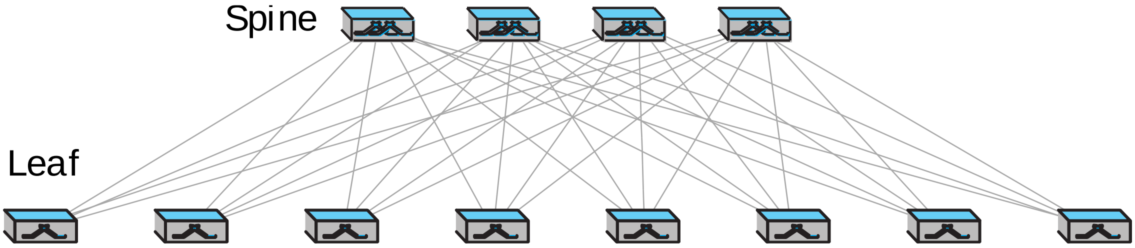

Figure 3. Leaf and Spine architecture with oversubscription rate 1:1.

3.3. N-HYPERCUBE

It is a graph-like topology, copying the structure of a hypercube of dimension *N* (also known as Q_n). The point here is that a switch is situated in every vertex of the *N*-hypercube chosen, also called nodes, and all of them are joined together according to the edges of the *N*-hypercube being used. Therefore, if a group of hosts are hanging on each of the switches, then any two given hosts being connected to any two switches will get together by virtue of following the shortest path within the *N*-hypercube between them [29]. Figure 4 depicts the simplest designs for *N*-hypercube.

It is to be noted that in this graph-like design, all switches have some hosts connected, as opposed to what happen with the tree-like designs, where only the lower layer have hosts attached. This fact makes that distances get reduced, although maintenance tasks and complexity get higher [30]. Furthermore, the nomenclature of the nodes of an *N*-hypercube helps us understand the possible paths between any two given switches, as the number of binary digits reflect the number of dimensions, meaning *N*, and any two given neighboring nodes differ in just one binary digit, stating that there is just one movement in that dimension to reach one another, given by swapping just one bit.

The dimension *N* is chosen to be the lowest available to admit the number of hosts involved in the design, and that amount is not tied to N in any way. It is to be noted that this topology is composed by 2^N nodes, being the switches, and $N·2^{N-1}$ edges, being the available links among those switches. Also, the maximum distance between any two given switches is *N*, being that the case of two nodes opposite each other.

Hence, any two given hosts may be hanging on the same switch, meaning they are just one hop away, or otherwise, all the way to N hops away. Furthermore, the nomenclature proposed above allows to easily see the distance between any two nodes, accounting for the number of ones found as the result of applying a logical XOR operator, those being the dimensions to get from source node to destination, accounting for the mismatching dimensions between two nodes. Additionally, the number of redundant paths between any two given hosts depends on the number of hops away, therefore, different load balancing policies may be put in place.

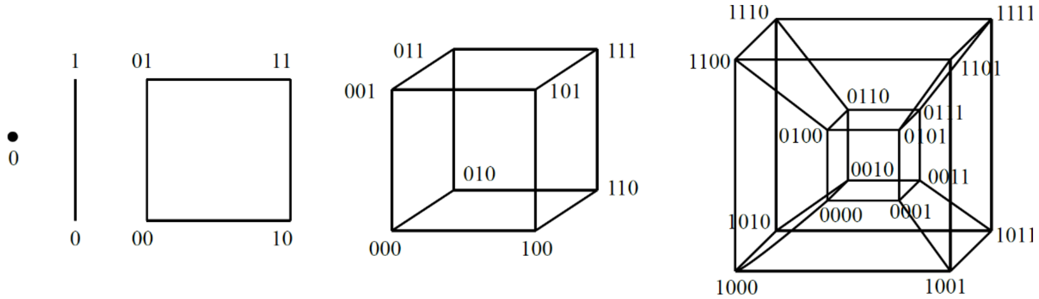

Figure 4. N-Hypercube architecture for N=0, N=1, N=2, N=3 and N=4 with oversubscription rate 1:1.

3.4. FOLDED N-HYPERCUBE

It is also a graph-like structure, improving the N-hypercube design with the establishment of a new link for each node towards its furthest node, being the opposite node. This way, the maximum distance between any two nodes gets reduced from *N* to *ceil(N/2)*, or $\lceil N/2 \rceil$, meaning the successive integer from N/2 in case that is not integer, getting shorter distances between further nodes, although getting a more complex interconnection network with more links to manage, as well as rising up costs [31]. It is to be noted the number of nodes within the topology is the same as above, this is, 2^N nodes, whereas the number of edges is $(N+1)·2^{N-1}$, being those the available links among those switches. Figure 5 exhibits the easiest designs for Folded *N*-hypercube.

Following the same nomenclature as the *N*-hypercube, the extra links go from a node referred with a binary number to its 1's complement. On the other hand, the way to calculate the possible paths from one source node to a destination node may be the same as described above [32]. Additionally, as Folded *N*-hypercubes (also known as FQ_n) add extra redundant paths to the design, they allow for the application of more load balancing policies. In summary, this extra feature allows for better performances and it is worthwhile in many case scenarios, although each one has to be assessed individually, as getting more interconnections may imply a worse manageability, which penalizes the implementation of full mesh topologies or other deployments with a great deal of links.

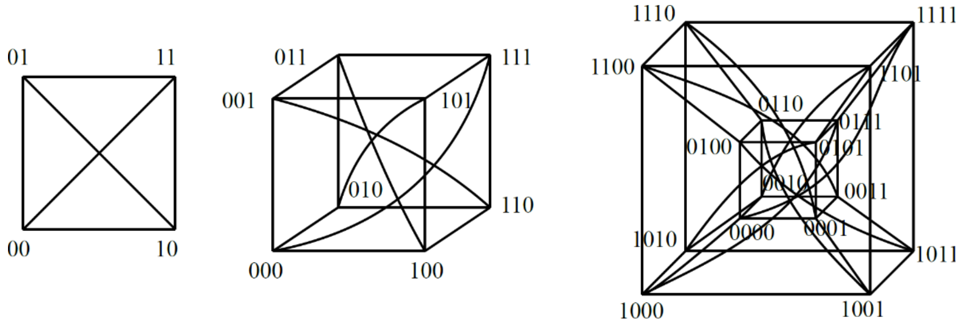

Figure 5. Folded N-Hypercube architecture for N=2, N=3 and N=4 with oversubscription rate 1:1.

4. DEFINING THE SCENARIOS

After having reviewed the main features of the four topologies presented, the following scenarios are going to be proposed so as to get the same number of hosts in each of them.

4.1. FAT TREE

In order to compare all four topologies proposed, some prerequisites have to be met in advanced. The most restrictive design is *Fat Tree*, as the number of switches (nodes) and links (edges) are determined by parameter *K*. It establishes the number of ports per switch to *K*, and even the number of hosts hanging on each switch, resulting in just *K/2*.

Therefore, this amount is going to be established for all scenarios in order to better compare their performance and their energy saving schemes. Focusing on Fat Tree, the number total of nodes, edges and hosts for each switch, Pod and the overall design of Fat Tree may be found in Table 1. The scenarios to be studied correspond to *K=4*, *K=8* and a generic *K* for Fat Tree, and in turn, the rest of topologies will be set up in order to have the same amount of hosts per switch, arranging the rest of nodes and edges accordingly.

Table 1. Number of items for the Fat Tree scenarios proposed.

Item Description	K=4	K=8	Generic value
Ports per switch	4	8	K
Hosts per switch	2	4	K/2
Hosts per Pod	4	16	$K^2/4$
Hosts overall	16	128	$K^3/4$
Edge switches	8	32	$K^2/2$
Aggreg. switches	8	32	$K^2/2$
Core switches	4	16	$K^2/4$
Switches overall	20	80	$5K^2/4$
Links overall	48	384	$3K^3/4$

4.2. LEAF AND SPINE

In this case, there is not such a parameter *K*, but for comparison purposes, let us suppose that the number of hosts per Leaf switch will be established to be *K/2*, whereas the rest of parameters are set as stated in Table 2, where the number of Leaf switches is sufficient to interconnect all hosts and the number of Spine switches is enough to bring together all Leaf switches. Therefore, on the one hand, for *K=4*, there will be 8 Leaf 4-port switches, with 2 hosts connected to each of them, as well as 2 Spine 8-port switches, providing a full mesh topology with all Leaf switches. On the other hand, for *K=8*, there will be 32 Leaf 8-port switches, with 4 hosts connected to each one, along with 8 Spine 32-port switches, giving full mesh topology. In order to work with generic expressions, let us consider parameter $L=log_2(K)$, which will be used in Table 2 for Leaf & Spine.

Table 2. Number of items for the Leaf and Spine scenarios proposed.

Item Description	K=4	K=8	Generic value
Equivalent value to K	L=2	L=3	L
Ports per Leaf sw	4	8	2^L
Ports per Spine sw	8	32	2^{2L-1}
Hosts per switch	2	4	2^{L-1}
Hosts overall	16	128	2^{3L-2}
Leaf switches	8	32	2^{2L-1}
Spine switches	2	8	2^{2L-3}
Switches overall	10	40	$5 \cdot 2^{2L-3}$
Links overall	32	192	2^{3L-1}

4.3. N-HYPERCUBE

Referring to this case, parameter *N* will be key in order to manage the topology and the number of switches and links may be found in Table 3, where *K=4* corresponds to *N=3* and *K=8* corresponds to *N=5* due to the total number of hosts in the topology, but this may be up to the designer.

Table 3. Number of items for the N-Hypercube scenarios proposed.

Item Description	K=4	K=8	Generic value
Equivalent value to K	N=3	N=5	N
Links to other sw	3	5	N
Ports per switch	5	9	2N-1
Hosts per switch	2	4	N-1
Hosts overall	16	128	$(N-1)2^N$
Switches overall	8	32	2^N
Links overall	12	80	$N \cdot 2^{N-1}$

It is to be remarked that if two hosts are connected to the same node, there is just one available path through that switch, meaning $\binom{N}{0}$. On the other hand, if they are one switch away, there may be $\binom{N}{1}$ possible switches, and if they are two switches away, the possible switches may be $\binom{N}{2}$, and so on. In summary, the sequence of numbers denoting the possible switches to be reached from an initial one is given by the sequence $a_n=\binom{N}{0},\binom{N}{1},\binom{N}{2},..,\binom{N}{N-1},\binom{N}{N}$, which result in the same values spotted at the i-th columns (i=0,1,..,N) of the Pascal's Triangle for row N, therefore, it might be expressed as $\vee_{i=0}^{N}\binom{N}{i}$.

Figure 5.a shows the Pascal's Triangle, where each cell is obtained by summing up its both upper cells, or otherwise, by calculating the combination of the N-th row and the i-th column, such as shown in (1).

$$\bigvee_{i=0}^{N}\binom{N}{i} = \bigvee_{i=0}^{N} \frac{N!}{N!\ \cdot\ (N-i)!} \qquad (1)$$

4.4. FOLDED N-HYPERCUBE

Regarding this case, it is an extension of the *N*-hypercube with the addition of extra links between any two opposite nodes, hence, there are more available paths to reach a certain node. As a consequence, *N* values selected are the same as above, although the number of switches and links are presented in Table 4.

Table 4. Number of items for the Folded N-Hypercube scenarios proposed.

Item Description	**K=4**	**K=8**	**Generic value**
Equivalent value to K	N=3	N=5	N
Links to other sw	4	6	N+1
Ports per switch	6	10	2N
Hosts per switch	2	4	N-1
Hosts overall	16	128	(N-1)2^N
Switches overall	8	32	2^N
Links overall	16	96	(N+1)· ·2^{N-1}

Regarding the amount of nodes to be reached from any given one with any number of hops away, it is greater than above due to the extra links. In order to undertake the calculations, it is to be considered the definition of factorial, this is, $! = n \cdot (n-1)!$, which yields to $(n-1)! = n!/n$. From this expression, it may be deduced that 0! = 1, and also that the factorial of a negative integer number is undefined, by means of (-1)!=0/0, which might be skipped if Gamma function is not considered. Therefore, factorial of negative numbers will be discarded.

The values required are just those shown in Figure 5.b, which may populate the Folded Pascal's Triangle for a hypercube of dimension *N* and the number of hops away are given by variable j, being the result of leaving the first column as it is, and summing up the second and the last value of a certain row, then the third and the before last, and so on. On the other hand, the same results may be obtained by the expression shown in (2).

$$\bigvee_{i=0}^{\left\lceil \frac{N}{2} \right\rceil}\left[\binom{N}{i} + \binom{N}{N-i+1}\right] : \left\lfloor \frac{i}{N-i+1} + 1 \right| \qquad (2)$$

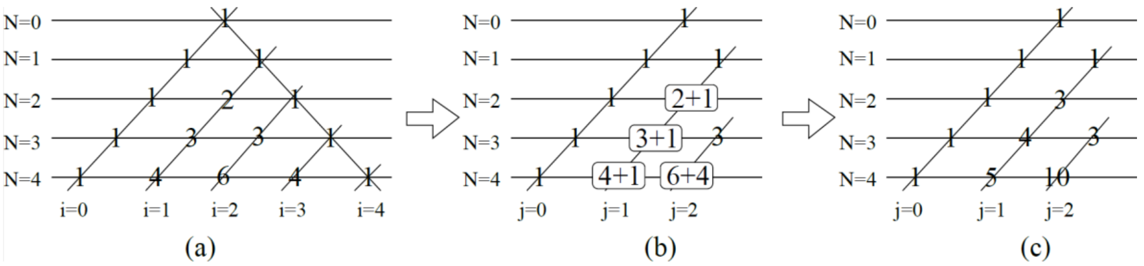

Figure 5. (a) Pascal's triangle for the first N values;
(b) how to fold the triangle leftwards;
(c) Folded Pascal's triangle for the first N values.

5. CALCULATIONS PERFORMED

All four topologies are going to be compared for three case scenarios, as proposed above, such that for 16 Hosts (2 per switch), 128 Hosts (4 per switch) and a generic value. The value presented for 0 switches means the one source host, which obviously does not take any switch and the value for 1 switch means the hosts hanging on the same switch as the source host (this is, 1 hop away). Then, it is important to distinguish between tree-like and graph-like topologies, as in the former, two or three hops away imply the use of 3 and 5 switches, respectively, due to the tree hierarchy, whereas in the latter, the number of hops away are just the number of switches away, due to the linear form of a graph.

It is also to be noted that in graph-like designs, the distance between any two switches holding a source and a destination host is the distance between those hosts minus one, as the source switch is not being counted in the way to the destination switch, as it is the starting point.

5.1.AVERAGE DISTANCE AMONG HOSTS, MEASURED IN SWITCHES

Here they are the results regarding the average distance counted in number of switches (*sw*) among any given pair of hosts (*ht).*

5.1.1. FAT TREE

In this topology, there is 1 host at a distance of 0 switches (this is, the host taken as a reference, which obviously does not need to pass through any switch to get to itself), the rest of the hosts hanging on the same Edge switch (being K/2 – 1 hosts) are at a distance of 1 switch, the rest of the hosts standing on the same Pod (being $K^2/4$ – K/2 hosts) are at a distance of 3 switches, as the path between them contains the source Edge switch, any Aggregate switch within that Pod and the destination Edge switch, and eventually, the rest of the hosts (being $K^3/4 – K^2/4$ hosts) are at a distance of 5 switches, as the path between them contains the source Edge switch, any Aggregation switch standing in the source Pod, a Core switch, any Aggregation switch standing in the destination Pod and the destination Edge. At this point, expression (3) focuses on K=4, whereas expression (4) does it on K=8, whilst expression (5) does it for a generic K value.

- Fat Tree (K=4) = $\frac{1\ host \cdot 0\ sw + 1\ host \cdot 1\ sw + 2\ hosts \cdot 3\ sw + 12\ hosts \cdot 5\ sw}{16\ hosts} = 3.81\ switches$ (3)

- Fat Tree (K=8) = $\frac{1\ host \cdot 0\ sw + 3\ hosts \cdot 1\ sw + 12\ hosts \cdot 3\ sw + 112\ hosts \cdot 5\ sw}{128\ hosts} = 4.68\ switches$ (4)

- Fat Tree (generic K) = $\frac{(K/2-1)\ ht\ \cdot\ 1\ sw\ +\ (K^2/4-K/2)\ ht\ \cdot\ 3\ sw\ +\ (K^3/4-K^2/4)\ ht\ \cdot\ 5\ sw}{K^3/4\ ht}$ (5)

5.1.2. LEAF AND SPINE

In this topology, there is 1 host at a distance of 0 switches (meaning the host taken as a reference, which clearly does not need to go through any switch to get to itself), the rest of the hosts connecting to the same Leaf switch (being $2^{L-1} – 1$ hosts) are at a distance of 1 switch whilst the rest of the hosts (being $K^3/4 – K^2/4$ hosts) are at a distance of 3 switches, as the path between them contains the source Leaf switch, any Spine switch and the destination Leaf switch. At this stage, expression (6) focuses on L=2, whereas expression (7) does it on L=3, whilst expression (8) does it for a generic L value.

- Leaf & Spine (L=2) = $\frac{\text{1 host} \cdot \text{0 sw} + \text{1 host} \cdot \text{1 sw} + \text{14 hosts} \cdot \text{3 sw}}{\text{16 hosts}}$ = 2.69 switches (6)

- Leaf & Spine (L=3) = $\frac{\text{1 host} \cdot \text{0 sw} + \text{3 hosts} \cdot \text{1 sw} + \text{124 hosts} \cdot \text{3 sw}}{\text{128 hosts}}$ = 2.92 switches (7)

- Leaf & Spine (generic L) = $\frac{(2^{L-1}-1)\ host\ \cdot\ 1\ sw\ +\ (2^{3L-2}-2^{L-1})\ hosts\ \cdot\ 3\ sw}{2^{\,3L-2}\ hosts}$ (8)

5.1.3. N-HYPERCUBE

In this topology, there is 1 host at a distance of 0 switches (being the host chosen as a reference, which evidently does not need to move through any switch to get to itself), the rest of the hosts linked to the same switch (being N – 2 hosts, as there are N – 1 hosts hanging on each switch) are at a distance of 1 switch, whereas the distance among the rest of the hosts will go from 2 to N+1, taking into account that there are as many switches at a certain distance as stated by the coefficients given by the Pascal Triangle regarding to row N, and each of those switches have N – 1 hosts hanging on. At this point, expression (9) focuses on N=3, whereas expression (10) does it on N=5, whilst expression (11) does it for a generic N value.

- N-Hypercube (N=3) = $\frac{1\ ht \cdot 0\ sw + 1\ ht \cdot 1\ sw + 6\ ht \cdot 2\ sw + 6\ ht \cdot 3\ sw + 2\ ht \cdot 4\ sw}{16\ ht} = 2.43\ sw$ (9)

- N-Hypercube (N=5) = $$\frac{\left[\begin{array}{l} 1\ ht \cdot 0\ sw + \ 3\ ht \cdot 1\ sw + 20\ ht \cdot 2\ sw + 40\ ht \cdot 3\ sw + \\ + 40\ ht \cdot 4\ sw + 20\ ht \cdot 5\ sw + \ 4\ ht \cdot 6\ sw \end{array}\right]}{128\ ht} = 3.49\ sw \qquad (10)$$

- N-Hypercube (generic N) = $$\frac{(N-2)\ ht\ \cdot 1\ sw + (N-1) \cdot \sum_{i=1}^{N+1} \binom{N}{i}\ ht \cdot (i+1)\ sw}{(N-1)\ \cdot 2^N\ ht} \qquad (11)$$

5.1.4. FOLDED N-HYPERCUBE

In this topology, there is 1 host at a distance of 0 switches (just the host chosen as a reference, which certainly does not need to traverse through any switch to get to itself), the rest of the hosts linked to the same switch (being N – 2 hosts, as there are N – 1 hosts hanging on each switch) are at a distance of 1 switch, whereas the distance among the rest of the hosts will go from 2 to $\lceil N/2 \rceil + 1$, taking into consideration that there are as many switches at a certain distance as stated by the coefficients given by the Folded Pascal Triangle regarding to row N, and each of those switches have N – 1 hosts hanging on. At this stage, expression (12) focuses on N=3, whereas expression (13) does it on N=5, whilst expression (14) does it for a generic N value.

- Folded N-Hypercube (N=3) = $$\frac{1\ ht \cdot 0\ sw + 1\ ht \cdot 1\ sw + 8\ ht \cdot 2\ sw + 6\ ht \cdot 3\ sw}{16\ ht} = 2.18\ sw \qquad (12)$$

- Folded N-Hypercube (N=5) = $$\frac{\left[\begin{array}{l} 1\ ht \cdot 0\ sw + \ 3\ ht \cdot 1\ sw + 24\ ht \cdot 2\ sw + \\ + 60\ ht \cdot 3\ sw + 40\ ht \cdot 4\ sw \end{array}\right]}{128\ ht} = 3.05\ sw \qquad (13)$$

- Folded N-Hypercube (generic N) = $$\frac{\left[\begin{array}{c} (N-2)\ ht\ \cdot 1\ sw + \\ +(N-1) \cdot \sum_{i=1}^{\lceil N/2 \rceil+1} \left[\binom{N}{i} + \binom{N}{N-i+1}\right] : \left\lfloor \frac{N}{N-i+1} + 1 \right\rfloor ht \cdot (i+1)\ sw \end{array}\right]}{(N-1)\ \cdot 2^N\ ht} \qquad (14)$$

5.2. AVERAGE DISTANCE AMONG END SWITCHES, MEASURED IN SWITCHES

Next, let us compare the average number of switches being used in a given transaction among switches which have hosts hanging on. Here they are the results regarding the average distance counted in number of switches (*sw*) among any given pair of end switches.

5.2.1. FAT TREE

In this topology, only the switches holding hosts are to be considered, hence, the distance between any pair of given Edge switches is going to be accounted. Therefore, there is 1 Edge switch at a distance of 0 switches (this is, the switch taken as a reference, which obviously does not need to pass through any other switch to get to itself), the rest of Edge switches hanging on the same Pod (being K/2 – 1 Edge switches) are at a distance of 2 switches, as the path between them contains any Aggregate switch within that Pod and the destination Edge switch, and eventually, the rest of the Edge switches (being $K^2/2$ – K/2 Edge switches) are at a distance of 4 switches, as the path between them contains any Aggregation switch standing in the source Pod, a Core switch, any Aggregation switch standing in the destination Pod and the destination Edge. At this point, expression (15) focuses on K=4, whereas expression (16) does it on K=8, whilst expression (17) does it for a generic K value.

- Fat Tree (K=4) = $$\frac{1\ edge \cdot 0\ sw + 1\ edge \cdot 2\ sw + 6\ edges \cdot 4\ sw}{} = 3.25\ switches \qquad (15)$$

$$8\ edges$$

- $$\text{Fat Tree (K=8)} = \frac{1\ edge \cdot 0\ sw + 3\ edges \cdot 2\ sw + 28\ edges \cdot 4\ sw}{32\ edges} = 3.69\ switches \qquad (16)$$

- $$\text{Fat Tree (generic K)} = \frac{((\mathrm{K}/2 - 1)\ edges\ \cdot\ 1\ sw\ +\ (\mathrm{K}^2/4 - \mathrm{K}/2)\ edges\ \cdot\ 3\ sw}{\mathrm{K}^2/2\ \text{edges}} \qquad (17)$$

5.2.2. LEAF AND SPINE

In this topology, only the switches holding hosts are to be considered, hence, the distance between any pair of given Leaf switches is going to be accounted. Hence, there is 1 Leaf switch at a distance of 0 switches (meaning the Leaf switch taken as a reference, which clearly does not need to go through any switch to get to itself), and the rest of the Leaf switches (being $2^{2L-1} - 1$ Leaf switches) are at a distance of 2 switches, as the path between them contains any Spine switch and the destination Leaf switch. At this stage, expression (18) focuses on L=2, whereas expression (19) does it on L=3, whilst expression (30) does it for a generic L value.

- $$\text{Leaf \& Spine (L=2)} = \frac{1\ leaf \cdot 0\ \text{sw} + 7\ leaves \cdot 2\ \text{sw}}{8\ leaves} = 1.75\ switches \qquad (18)$$

- $$\text{Leaf \& Spine (L=3)} = \frac{1\ leaf \cdot 0\ \text{sw} + 31\ leaves \cdot 2\ \text{sw}}{32\ leaves} = 1.94\ switches \qquad (19)$$

- $$\text{Leaf \& Spine (generic L)} = \frac{(2^{2\mathrm{L}-1} - 1)\ leaves\ \cdot\ 2\ sw}{2^{\ 2L-1}\ leaves} \qquad (20)$$

5.2.3. N-HYPERCUBE

In this topology, all switches are to be accounted in this point as all of them hold hosts, thus, all of them are end switches (*es*). Hence, there is 1 switch at a distance of 0 switches (being the host chosen as a reference, which evidently does not need to move through any switch to get to itself), whereas the distance among the rest of the switches will go from 1 to N, taking into account that there are as many switches at a certain distance as stated by the coefficients given by the Pascal Triangle regarding to row N. At this point, expression (21) focuses on N=3, whereas expression (22) does it on N=5, whilst expression (23) does it for a generic N value.

- $$\text{N-Hypercube (N=3)} = \frac{1\ es \cdot 0\ sw + 3\ es \cdot 1\ sw + 3\ es \cdot 2\ sw + 1\ es \cdot 3\ sw}{8\ es} = 1.50\ sw \qquad (21)$$

- $$\text{N-Hypercube (N=5)} = \frac{\begin{bmatrix} 1\ es \cdot 0\ sw + \ \ 5\ es \cdot 1\ sw + 10\ es \cdot 2\ sw + \\ + 10\ es \cdot 3\ sw + \ \ 5\ es \cdot 4\ sw + \ \ 1\ es \cdot 5\ sw \end{bmatrix}}{32\ es} = 2.50\ sw \qquad (22)$$

- N-Hypercube (generic N) = $$\frac{\sum_{i=0}^{N}\binom{N}{i}\, es \cdot i\ sw}{2^{N}\, es} \qquad (23)$$

5.2.4. FOLDED N-HYPERCUBE

In this topology, all switches are to be accounted in this point as all of them hold hosts, thus, all of them are end switches (*es*). Therefore, there is 1 switch at a distance of 0 switches (just the host chosen as a reference, which certainly does not need to traverse through any switch to get to itself), whilst the distance among the rest of the switches will go from 1 to $\lceil N/2 \rceil$, taking into consideration that there are as many switches at a certain distance as stated by the coefficients given by the Folded Pascal Triangle regarding to row N. At this point, expression (24) focuses on N=3, whereas expression (25) does it on N=5, whilst expression (26) does it for a generic N value.

- Folded N-Hypercube (N=3) = $$\frac{1\ es \cdot 0\ sw + 4\ es \cdot 1\ sw + 3\ es \cdot 2\ sw}{8\ es} = 1.25\ sw \qquad (24)$$
- Folded N-Hypercube (N=5) = $$\frac{1\ es \cdot 0\ sw + 6\ es \cdot 1\ sw + 15\ es \cdot 2\ sw + 5\ es \cdot 3\ sw}{32\ es} = 1.59\ sw \qquad (25)$$
- Folded N-Hypercube (generic N) = $$\frac{\sum_{i=0}^{\lceil N/2 \rceil}\left[\binom{N}{i} + \binom{N}{N-i+1}\right] : \left\lfloor \frac{N}{N-i+1} + 1 \right\rfloor es \cdot i\ sw}{2^{N}\ sw} \qquad (26)$$

5.3. AVERAGE NUMBER OF SWITCHES IN ENERGY SAVING MODE

It is to be considered that switches not being in use in a certain interval of time allow for the application of energy saving schemes, in a way that unused switches may be put in idle state through control signals generated by an orchestrator, thus saving resources. In order to undertake those calculations, the results obtained before are being used. Those results are summarized in Table 5.

Table 5. Average distance among switches in the scenarios proposed.

Topology	Total Switches		Total Links		Avg. dist. among Hosts		Avg. dist. among Switches	
	K=4	K=8	K=4	K=8	K=4	K=8	K=4	K=8
Fat Tree	20	80	48	384	3.81	4.68	3.25	3.69
Leaf Spine	10	40	32	192	2.69	2.92	1.75	1.94
N-Hyperc.	8	32	12	80	2.43	3.49	1.50	2.50
Folded N-Hy.	8	32	16	96	2.18	3.05	1.25	1.59

Eventually, the average idle rate of switches is calculated by dividing the total switches except its rounded average distance among Hosts and the total switches, which accounts for a rate between 0 and 1, the former meaning that all accounted switches are busy, whereas the latter meaning that all accounted switches are idle. Therefore, the closer to a value of 1 implies that there are more switches in idle state, hence, the application of energy saving policies would make for better performances regarding energy consumption [33]. Those results get summarized in Table 6.

Table 6. Average idle rate of switches in the scenarios proposed.

Topology	Total Switches		Avg.dist.Hosts		Rounded Avg. dist.Hosts		Avg Idle Rate of Switches	
	K=4	K=8	K=4	K=8	K=4	K=8	K=4	K=8
Fat Tree	20	80	3.81	4.68	20-4=16	80-5=75	16/20	75/80
Leaf Spine	10	40	2.69	2.92	10-3=7	40-3=37	7/10	37/40
N-Hyperc.	8	32	2.43	3.49	8-3=5	32-4=28	5/8	28/32
Folded N-Hy.	8	32	2.18	3.05	8-2=6	32-3=29	6/8	29/32

It is to be considered that Data Centers may have different duty cycles, depending on the numbers of active users at any point of time and the workload being managed by those users. Therefore, it may be interesting to implement sleeping mechanisms in environments where the average use is not very high, as the implementation of energy saving policies may make a difference. In that case, the sleeping policies put in place for switches being idle for a given period of time may need a fast recovery time in order to be able to quickly get back up again when new traffic may need to use such switches.

6. CONCLUSION

In this paper, different Data Center topologies have been presented, such as Fat Tree, Leaf and Spine, *N*-Hypercube and Folded *N*-Hypercube. On the one hand, the first and the second ones are considered as tree-like topologies, where there are different rows of switches forming a hierarchy, such that just the lower layer holds the different hosts hanging on them. On the other hand, the third and the fourth ones are branded as graph-like topologies, where all switches have hosts connected to them.

For each of those topologies, three case scenarios have been defined, where the number of hosts has been fixed for the first and second scenarios, leaving a third one with generic values.

The measurements obtained are the average distance among end switches (being those switches with hosts hanging on) and among hosts (being those the end devices connected to end switches), both measured in number of intermediate switches to reach destination, along with the average idle rate of switches (being those not taking part in a given transaction), where all calculations have been undertaken on an arithmetic manner.

It is to be noted that the measurements about idle rate in switches seem quite high in all cases, which may bring about the idea of implementing energy saving schemes for the idle switches in all of those topologies so as to be more energy efficient, thus saving both electric power and cooling resources.

On the other hand, the calculations performed show that there is no such a better topology, as results vary depending on the number of items involved, and furthermore, other key performance indicators may be seen, such as the complexity of the network because of the amount of interconnections.

Additionally, in networks with the need of steady rates of latency and jitter, such as VoIP or video streaming, tree-like architectures make the difference, taking into account that Leaf and Spine seems the best, but it does not scale good, whereas Fat Tree does achieve a good trade off in all circumstances.

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.

Carlos JUIZ: Analyzed the results.

CONFLICT OF INTEREST

There is no conflict of interest in this study.

REFERENCES

[1] Mao Y, You C, Zhang J, Huang K, Ketaief K B. A Survey on Mobile Edge Computing: The Communication Perspective. ***IEEE Communications Surveys & Tutorials***, 19, 4, 2322-2358, (2017).

[2] Ai Y, Peng M, Zhang K . Edge computing technologies for Internet of Things: a primer. ***Digital Communications and Networks***, 4, 2, 77-86, (2018).

[3] Okwuibe J, Ahmad I, Yianttila M, Liyanage M. Cloud and MEC security. ***A Comprehensive Guide to 5G Security***, 373-397, (2018).

[4] Prabhu C S R . Overview - Fog Computing and Internet-of-Things (IoT). ***EAI Endorsed Transactions on Cloud Systems***, 10, 5, 1-24. China Communications Magazine Co. Ltd, Beijing, (2017).

[5] Habibi P et al . Fog Computing: A Comprehensive Architectural Survey. ***IEEE Access***, 8, 69105-69133, (2020).

[6] Waqas M et al . Mobility-Aware Fog Computing in Dynamic Environments: Understandings and Implementation. ***IEEE Access***, 7, 38867-38879, (2018).

[7] Mukherjee M, Gorlatova M, Gross J, Aazam M. Sustainable Infrastructures, Protocols and Research Challenges for FOG Computing. ***IEEE Access***, 8, 110943-110946, (2020).

[8] S. Vikram . Green Computing.***International Conference on Green Computing and Internet of Things***, 1, 767-772, (2015).

[9] B. Wadhwa, A. Verma . Carbon efficient VM placement and migration technique for green federated cloud datacenters. ***International Conference on Advances in Computing, Communications and Informatics***, 17041923, 2297-2302, (2014).

[10] R. Bala and J. Mann . A Research Paper on Green Computing Using Energy Efficient Task Allocation Strategy in Cloud Environment. ***International Journals of Advanced Research in Computer Science and Software Engineering***, 31907523, 186-191, (2017).

[11] T. Wang, Z. Su, Y. Xia, M. Hamdi . Rethinking the Data Center Networking: Architecture, Network Protocols, and Resource Sharing. ***IEEE Access***, 2, 1481-1496, (2014).

[12] C. Koronen, M. Åhman, L.J. Nilsson . Data centres in future European energy systems—energy efficiency, integration and policy. ***Energy Efficiency Journal***, 13, 129–144, (2020).

[13] Mahmud R, Ramamohanarao K, Buyya R . Latency-Aware Application Module Management for Fog Computing Environments. ***ACM Transactions on Internet Technology***, 19, 1, 9, 1-21, (2019).

[14] Kamath R . Fog Computing – a practical overview. ***International Journal of Innovative Research in Computer and Communication Engineering***, 5, 4, 7819-7822, (2017).

[15] Filiposka S, Mishev A, Juiz C . Community-base VM placement framework.***The Journal of Supercomputing***, 71, 12, 4504-4528. Springer-Heidelberg, (2015).

[16] Roman R, Lopez J, Mambo M . Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. ***Future Generation Computer Systems***, 78, 2, 680-698, (2018).

[17] Sabella D, Vaillant A, Kuure P, Rauschenbach U . Mobile-Edge Computing Architecture: The role of MEC in the Internet of Things. ***IEEE Consumer Electronics Magazine***, 5, 4, 84-91, (2016).

[18] Sethi P, Sarangi S R . Internet of Things: Architectures, Protocols, and Applications. ***Journal of Electrical and Computing Engineering***, 1, 1-25, (2017).

[19] Porambage P. et al . Survey on Multi-Access Edge Computing for Internet of Things Realization. ***IEEE Communications Survey & Tutorials,*** 20, 4, 2961-2991, (2018).

[20] Kaur P, Rani A . Virtual machine migration in Cloud computing. ***International Journal of Grid Distribution Computing***, 8, 5, 337-342, (2015).

[21] Roig P J, Alcaraz S, Gilly K, Juiz C . Algebraic modelling of a generic Fog scenario for moving IoT devices. ***Lecture Notes in Networks and Systems*** (Springer), in press, (2021).

[22] Osanaiye O, Chen S, Yan Z, Lu R, Choo K R, Dlodlo M. From Cloud to Fog Computing: A Review and a Conceptual Live VM Migration Framework. ***IEEE Access***, 5, 8284-8300 (2017).

[23] Roig P J, Alcaraz S, Gilly K . Arithmetic Study on Energy Saving for some common Data Centre Topologies. ***Proceedings of the 8th European Conference ECRES 2020***, Istanbul (Turkey), 744-751. Published without a DOI, (2020).

[24] Al-Fares M, Loukissas A, Vahdat A . A scalable, commodity data center network architecture.***ACM SIGCOMM Computer Communication Review***, 38, 4, 63-74 (2008).

[25] Roig P J, Alcaraz S, Gilly K, Juiz C . Modelling VM Migration in a Fog Computing Environment.***Journal Elektronika ir Elektrotechnika***, 25, 5, 75-81 (2019).

[26] Okafor K C . Leveraging Fog Computing for scalable IoT datacenter using Spine-Leaf network topology. ***Journal of Electrical and Computer Engineering***, 1, 1-11 (2017).

[27] Roig P J, Alcaraz S, Gilly K, Filiposka S, Aknin N . Formal Algebraic Specification of an IoT/Fog Data Centre for Fat Tree or Leaf and Spine Architectures. ***Proceedings of International Conference on Electrical, Communication and Computer Engineering***, 1-6 (2020).

[28] Roig P J, Alcaraz S, Gilly K, Juiz C . Modelling a Leaf and Spine Topology for VM Migration in a Fog Computing Environment. ***Proceedings of 24th International Conference ELECTRONICS 2020***, 1-5, (2020).

[29] Harary F, Hayes J P, Wu J-H . A survey of the theory of hypercube graphs. ***Computer and Mathematics with Applications***, 15, 4, 277-289 (1988).

[30] Roig P J, Alcaraz S, Gilly K, Juiz C . Modelling a Plain N-Hypercube Topology for migration in Fog Computing. ***Lecture Notes in Electrical Engineering*** (Springer), in press, (2021).

[31] Zhu Q, Xu J-M, Hou X, Xu M . On reliability of the folded hypercubes. ***Journal of Information Sciences***, 177, 8, 1782-1788 (2007).

[32] Roig P J, Alcaraz S, Gilly K, Juiz C . Modelling a Folded N-Hypercube Topology for migration in Fog Computing. ***Lecture Notes in Electrical Engineering*** (Springer), in press, (2021).

[33] Agarwal, U., Jain, N. Distributed Energy Resources and Supportive Methodologies for their Optimal Planning under Modern Distribution Network: a Review. ***Technology and Economics of Smart Grids and Sustainable Energy***, 4, 3, 1-21 (2019).