LaScaDa: A Novel Scalable Topology for Data Center Network

Zina Chkirbene, Rachid Hadjidj, Sebti Foufou, *Member, IEEE*, and Ridha Hamila, *Senior Member, IEEE*

***Abstract*— The growth of cloud-based services is mainly supported by the core networking infrastructures of large-scale data centers, while the scalability of these services is influenced by the performance and dependability characteristics of data centers. Hence, the data center network must be agile and reconfigurable in order to respond quickly to the ever-changing application demands and service requirements. The network must also be able to interconnect the big number of nodes, and provide an efficient and fault-tolerant routing service to upper-layer applications. In response to these challenges, the research community began exploring novel interconnect topologies, namely: Flecube, DCell, Ficonn, HyperFlaNet and BCube. However, these topologies either scale too fast (grows exponentially in size), or too slow, and therefore suffer from performance bottlenecks. In this paper, we propose a novel data center topology called LaScaDa (Layered Scalable Data Center) as a new solution for building scalable and cost-effective data center networking infrastructures. The proposed topology organizes nodes in clusters of similar structure, then interconnect these clusters in a well-crafted pattern and system of coordinates for nodes to reduce the number of redundant connections between clusters, while maximizing connectivity. LaScaDa forwards packets between nodes using a new hierarchical row-based routing algorithm. The algorithm constructs the route to the source based on the modular difference between the source and destination coordinates. Furthermore, the proposed topology interconnects a large number of nodes using a small node degree. This strategy increases the number of directly connected clusters and avoids redundant connections. As a result, we get a good quality of nodes in terms of average path length (APL), bisection bandwidth, and aggregated bottleneck throughput. Experimental results show that LaScaDa has better performance than DCell, BCube, and HyperBcube in terms of scalability, while providing a good quality of service.**

***Index Terms*— Data center network, network topology, average path length, bisection bandwidth.**

I. Introduction

MASSIVE data centers are being built around the world to provide various cloud computing services such as online office, online social networking, Web search, and IT infrastructure out-sourcing [1]. For instance, Microsoft, IBM, Google, Amazon, Yahoo and eBay are running data centers with at least 50,000 nodes (servers)[1] for each one of them [2], [3]. Mega data centers provide the core support infrastructure for the cloud and amounts for up to 45% of the total implementation cost. Consequently, the data center infrastructure must be well designed to maintain the cost of both deployment and maintenance at an acceptable level [4]. In addition, data availability and scalability are considered as critical parameters in the design of a data center topology because of their big impact on the infrastructure cost. In fact, according to industry estimates, the US data center market increased by 23 USD billions between 2005 and 2009, growing from 16.2 USD billions in 2005 to almost 39 USD billions in 2009 [5]. Moreover, data centers networking is traditionally built around Top of Rack (ToR) switches interconnected through End of Rack (EoR) switches, which are in turn connected through core switches. Consequently, this approach is very costly, while leading to significant bandwidth oversubscription towards the network core. All these issues encouraged several researchers to propose new topologies for scalable and cost-effective network infrastructures, namely: FatTree [6], FiConn [7], DCell [8], BCube [9], and HyperBcube [10].

However, proposed topologies suffer from performance bottlenecks and costly implementations [11]. In fact, with switches having a small port count n, the number of nodes in BCube increases only by a factor of n, leading to a potentially high number of recursive layers for large-scale data centers. For instance, with $n = 4$, 6 layers are required to construct a data center with $4^6 = 4096$ nodes. A 6-layer BCube network requires 6 interface cards for each node which are expensive and difficult to manage in practice. Hence, BCube can suffer from scalability issues when employing cost-effective small port count switches and a small node degree. With DCell topology, a data center network having millions of nodes can be easily constructed with a node degree of 4. However, its major drawback lies in its high wiring complexity, and inefficient local re-routing algorithm (the full connection in each network layer makes distances between pairs of nodes in different layers very long) [12]. So, DCell reduces the diameter of the entire network, but increases the wiring complexity, which makes the deployment of a DCell data center very complicated [13].

In this work, we propose a novel interconnection network topology called LaScaDa (Layered Scalable Data Center) that scales faster than HyperBcube, BCube, Flecube, and DCell,

Manuscript received January 22, 2017; revised January 30, 2018, August 17, 2018, and February 17, 2020; accepted June 9, 2020; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor K. Argyraki. Date of publication August 3, 2020; date of current version October 15, 2020. This work was supported by the Qatar National Research Fund (a member of Qatar Foundation) under NPRP Grant 6-718-2-298, by Qatar University Internal Grant IRCC-2020-001, and by Qatar National Library. *(Corresponding author: Zina Chkirbene.)*

Zina Chkirbene and Ridha Hamila are with the College of Engineering, Qatar University, Doha 2713, Qatar (e-mail: zina.chk@gmail.com; hamila@qu.edu.qa).

Rachid Hadjidj is with the Ahmed Bin Mohammed Military College (ABMMC), Doha 22988, Qatar (e-mail: rhadjidj@qu.edu.qa).

Sebti Foufou is with the LIB Laboratory, University of Burgundy, 21000 Dijon, France (e-mail: sfoufou@qu.edu.qa).

Digital Object Identifier 10.1109/TNET.2020.3008512

[1]In this document we will use the words "node" and "server" interchangeably.

while enjoying a low average path length, high bisection bandwidth, high aggregate bottleneck throughput, and more importantly high scalability using the same number of switches and links per node as HyperBcube and BCube. We also propose fault-free and fault-tolerant routing algorithms to be used in the case of absence or presence of network failures respectively.

The followings are some of the contributions proposed in this paper:

1) A novel data center topology called LaScaDa, capable of scaling the entire network to millions of nodes using nodes with small degrees and small port count switches. While using the same number of links and switches per node as HyperBcube and Bcube, LaScaDa outperforms these topologies in terms of Scalability, Average Path Length (APL), Bisection bandwidth and Aggregated Bottleneck Throughput (ABT).
2) A new physical structure algorithm to interconnect nodes. Nodes are organized in clusters of similar structure, which are then interconnected via switches using a well-crafted pattern and system of coordinates for nodes. The objective is to reduce the number of redundant connections between clusters while maximizing connectivity. The algorithm computes the size of the linked clusters set for each possible switch, then selects the switch that maximizes the number of connected clusters.
3) A fault-free routing algorithm to route data between nodes. The algorithm determines the route between nodes based on the modular difference between the coordinates of the source and destination. Given multiple routing paths for a source, the shortest one is selected.
4) A fault-tolerant routing algorithm to be used in the case of link failure. The algorithm changes the routing table of some nodes and looks for new intermediate nodes to forward a packet to its destination.

The rest of this paper is organized as follows: Section 2 provides a summary of relevant related work in the field. Sections 3 and 4 describe the proposed connection patterns and key features of LaScaDa. Experimental results are presented in Section 5 followed by a discussion in Section 6. Section 7 concludes the paper.

II. Related Work

Different topologies for data centers have been proposed in the literature and can be classified into two categories: Switch-Centric and Server-Centric.

A. The Switch-Centric Category

The Switch-Centric category uses intelligent switches for a smart routing of packets in a data center. Some data center topologies in this category are VL2 [14], Clos Network [15], FatTree [16], [17], JellyFish [18], DOS [19] and Hypac [20].

- VL2 has been proposed as a solution for some critical issues in conventional data centers. By exploiting a uniform high capacity traffic from node to node, VL2 overcomes some critical issues such as Oversubscription, Agility and Fault-tolerance. In addition, and by employing virtual machines, VL2 improves network availability even with hardware or link failures. However, VL2 uses Valiant Load Balancing (VLB) that before forwarding a packet, selects randomly an intermediate switch. This is impractical, especially in the case where two hosts connected to the same edge switch want to communicate.
- Clos Network [15] is a multi-rooted tree consisting of three levels of switches: Top of the Rack (ToR), aggregation and intermediate switches. The intermediate and aggregation switches must have the same number of ports, however, the number of ports on a ToR switch is not limited. The number of ports on intermediate switches is used to compute the number of switches in the network. If each switch on the intermediate level has n ports, the topology uses n aggregation switches and $\frac{n}{2}$ intermediate switches. Only one link is used to connect an intermediate switch to an aggregation switch. The remaining $\frac{n}{2}$ ports on each aggregation switch is connected to $\frac{n}{2}$ different ToR switches. The number of connected nodes in a Clos Network topology is equal to $n_{ToR} \times \frac{n^2}{4}$ where n_{ToR} is the number of ports on each ToR switch. Also, server-to-server latency varies depending on the traffic path used.
- FatTree [16] is a special case of a Clos network. Fat-tree topology consists of a core and pods. The core is composed of switches that interconnect the pods. A pod is composed of aggregation switches, edge switches, and servers. Each port of each switch in the core is connected to a different pod through an aggregation switch. Within a pod, aggregation switches are connected to all edge switches. Finally, each edge switch is connected to a different set of servers. Unlike Tree topologies, all switches at all levels in FatTree are of the same type. High-performance switches are not necessary in the Aggregate and Core levels. However, the number of nodes in FatTree is limited by the number of switch ports.Fatree is considered a special instance of Clos topologies where the number of top of racks is equal to n [21].
- Jellyfish [18] increases exponentially the number of nodes in a data center. It is constructed based on a random graph at the ToR switch layer. Each ToR switch has n ports, where r of them are used to connect it to other ToR switches while remaining ports are used for nodes interconnection. So, for a network with N racks, when the same number of ports are deployed by every switch, Jellyfish connects $N \times (n-r)$ nodes. The average path length in Jellyfish is shorter than Fat-tree, while the diameter is at least the same.

B. The Server-Centric Category

In addition to intelligent switches, the Server-Centric category uses also servers able to forward packets [22]. In this category, several topologies are used such as DCell [8], BCube [9], HyperBcube [10], Flecube [23] and FiConn [7].

- DCell has a recursive structure, where the basic element is called DCell$_0$. The switch in DCell$_0$ connects all of its nodes. In DCell$_k$, each node has $k+1$ links: the first link (or level$_0$ link) is connected to a switch when forming DCell$_0$, and level$_i$ link is connected to a node in the same DCell$_i$. Most of DCell nodes act as routers and are equipped with multiple interface cards (NICs). Only computational nodes are considered as routers.

As a result, DCell topology scales double exponentially because of additional and long communication wiring links, switches, and nodes.

- BCube is a Server-Centric network structure [9]. BCube makes use of more switches when constructing a higher-level topology. BCube$_1$ is constructed from n BCube$_0$ and n n-port switches. More generally, BCube$_k$ is built from n BCube$_{k-1}$ and n^k extra n-port switches connected to exactly one node in each BCube$_{k-1}$. When constructing higher level structures, BCube requires more switches compared with DCell which uses only level$_0$ n-port switches. However, both topologies require servers to have (k+1) NICs. The result is that servers will be involved in switching more packets in DCell than in BCube.
- FiConn has a recursive structure [7], where a high-level FiConn is built using low-level FiConns. FiConn uses only existing backup port on each node for interconnection, while no extra hardware cost is introduced. This topology provides some improvements over FatTree and uses the interconnection intelligence on nodes rather than on switches, hence, reducing the number of switches. If we denote the total number of nodes by N, and use n-port switches to connect them, then the number of switches needed in FatTree is $\frac{5N}{n}$ (2 edges, 2 aggregated and 1 core for each pod), while FiConn requires only $\frac{N}{n}$ switches. Therefore, FiConn reduces the cost of switches by 80 % compared with FatTree [16].
- Flecube is a topology built using nodes equipped with multiple ports directly connected to other nodes via bidirectional communication links, without using any switches [23]. In Flecube, a higher-level Flecube is constructed from lower-level Flecubes. For instance, level-1 Flecube is constructed using $k+1$ n-port servers connected by means of a complete graph via ports in group 1. Flecube uses servers equipped with multiple ports to construct its network without using any switches or routers. So, as the number of connected servers increases, network performance and data transfer rates decrease. Therefore, servers equipped with multiple ports do not allow much scalability, while traffic problems and packet collisions tend to arise.
- HyperBcube is a recursive topology [10]. The first layer of the HyperBcube contains n servers and one n-port switch. Starting from the second layer ($k \geq 2$), HyperBcube can be considered as an $n^2 \times n^{(2\times k-3)}$ matrix having $n^{(2\times k-3)}$ columns, where each column contains exactly n^2 servers belonging to a n^2 (k-1)-layer HyperBcube. A column-based connection is used to connect the n^2 servers located at the same column by using exactly n n-port switches. However, the connection pattern in HyperBcube is inefficient since it results in redundant cluster connections. In addition, if two clusters do not have any intermediate switch, 8 hops are needed to connect servers in these clusters.

III. LaScaDa Topology

A. Motivation

Data centers provide services for cloud computing and therefore can be quite big. For instance, Microsoft Live Online

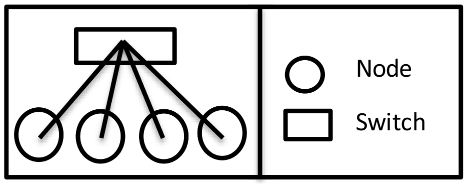


Fig. 1. β link for data center connection.

Services data center is one of the largest data centers, and spans for more than 700,000 square feets [24]. Hence, the data center infrastructure must be agile and cost-effective to support the ever-growing critical cloud needs in terms of computation, storage, and applications. In particular, the data center network topology must be well designed. Scalability is an important factor in this regard due to its significant impact on network performance [25]. The scalability of a topology is given by a formula that determines the exact number of nodes that need to be connected given some input parameters such as the number of ports per switch, and some structural parameters such as the number of layers. A topology is said to scale fast if the difference between allowed numbers of nodes grows very fast. Table I shows a classification of some existing topologies based on their scalability, bisection bandwidth, and diameter [8], [9], [14], [16]. In the table, k and n are the number of ports per node and the number of ports per switch respectively.

Table I reveals that FatTree and VL2 have some physical limitations to scale up. In fact, if we denote by n the maximum port count of switches, VL2 (even with a three-layer network) can only connect $\frac{n^3}{4}$ nodes, which is insufficient for a large-scale data center. DCell and BCube provide good scalability, however, DCell has a high wiring complexity and BCube requires more than three layers to scale up to a large size. For instance, with a 4-port switch, we need five layers to build a data center with $4^5 = 1024$ nodes. A 5-layer BCube network needs five interface cards per node, which is obviously expensive and difficult to manage in practice. Hence, BCube has scalability issues when employing cost-effective small degree nodes and small port count switches.

A data center network consists of nodes, switches, and links. Basically, there are three types of links: α (linking two nodes), β (linking a node and a switch) and γ (linking two switches). The β connection is considered as one of the most efficient connection as it provides multiple non-blocking paths that allow multiple pairs of nodes to share their communication channels.

All the previously mentioned limitations have been considered in the design of our proposed topology LaScaDa which scales up a data center to millions of nodes while providing good quality of services. LaScaDa uses a new physical structure and routing algorithms. Using exclusively β links and small port count switches, the proposed topology increases the number of directly connected clusters per layer, while avoiding redundant cluster connections. Hence, LaScaDa enjoys low APL and latency while increasing the number of nodes.

B. Some Definitions

1) Cluster: A cluster is a group of n servers connected to an "external" n-port switch (See Figure 2).

TABLE I

SOME DATA CENTER NETWORK TOPOLOGIES AND THEIR SCALABILITY, BISECTION BANDWIDTH, AND DIAMETER

	FatTree	VL2	DCell	Ficonn	BCube
Scalability (S)	$\frac{n^3}{4}$	$5 \times n^2$	$((n+\frac{1}{2})^{2^k} - \frac{1}{2}, (n+1)^{2^k} - 1)$	$\geq ((\frac{n}{4})^{2^k} 2^{k+2})$	n^k
Bisection Bandwidth	$\frac{n^3}{8}$	$10n^2 - 20$	$\frac{S}{4 \log_n S}$	$> \frac{S}{4 \times 2^k}$	$\frac{S}{2}$
Diameter	6	6	$> 2^k - 1$	$> 2^k - 1$	k

TABLE II

NUMBER OF NODES UNDER DIFFERENT CONFIGURATIONS

n	Tree-based Topology			k	Recursive Topology				
	Basic Tree	Fat Tree	Clos Network		DCell	BCube	Ficonn	HyperBcube	LaScaDa
4	9	3	3	2	20	16	16	64	32
				3	420	64	32	1024	4096
				4	176820	252	64	16384	131072
6	64	16	8	2	42	36	81	216	648
				3	1806	216	822	7776	69984
				4	3×10^6	1269	42×10^6	823543	7×10^6
8	216	54	36	2	72	64	256	512	2048
				3	5252	512	8192	32768	524288
				4	27×10^6	4096	4×10^6	2×10^6	134×10^6
16	512	128	96	2	272	256	4096	4096	32468
				3	74256	4096	2×10^6	1×10^6	67×10^6
				4	5514×10^6	65536	274×10^6	268×10^6	137438×10^6

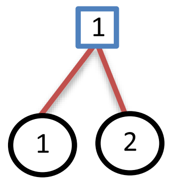


Fig. 2. A cluster example when n=2.

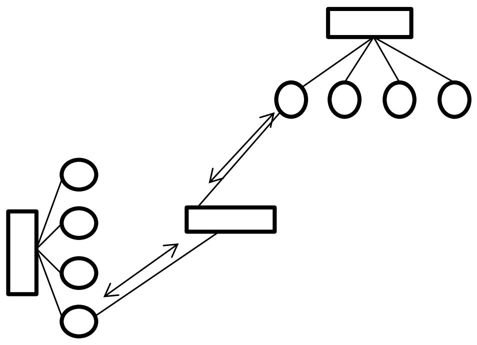


Fig. 3. An example of directly connected clusters.

2) Directly Connected Clusters: Two clusters are said to be directly connected if there is at least one switch that connects them directly (See Figure 3).

3) Not Directly Connected Clusters: Two clusters are said to be not directly connected if two or more switches need to be traversed to find a route from one to the other (see Figure 4). Note that reducing the number of not directly connected clusters reduces the average path length.

C. Physical Structure

A LaScaDa network built out of n-port switches is a layered and recursive topology such that a k-layer LaScaDa network ($k > 1$) is built by interconnecting $\frac{n^3}{2}$ $(k-1)$-Layer LaScaDa networks using n-port switches. These switches are qualified as *internal switches*. The starting building block in LaScaDa is 1-layer LaScaDa, also called cluster. A cluster consists of n nodes connected to one n-port switch (see Figure 2). For a k-layer LaScaDa where $k > 1$, the $\frac{n^3}{2}$ $(k-1)$-Layer laScaDa are interconnected in a well crafted pattern that maximizes the number of connections between clusters, while avoiding redundant connections. In what follows we will explain this pattern by showing how to build a 2-layer LaScaDa network from 1-layer LaScaDa networks.

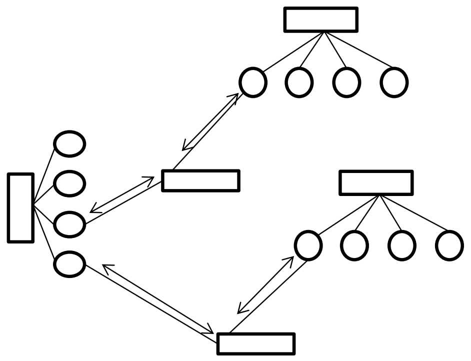


Fig. 4. An example of not directly connected clusters.

A 2-layer LaScaDa is built out of $\frac{n^3}{2}$ 1-layer LaScaDa networks (clusters) numbered from 1 to $\frac{n^3}{2}$, interconnected with $\frac{n^3}{2}$ n-port switches (internal switches) numbered from 1 to $\frac{n^3}{2}$. We perform the interconnection in an iterative process for $i = 1$ *to* n, where n is the number of ports per switch. In each iteration i we: (1) connect node i of the first cluster to the first internal switch j $(j = 1..\frac{n^3}{2})$ that allows to connect cluster 1 to the maximum number of other clusters; (2) then we connect each node i of each other cluster m $(m = 2..\frac{n^3}{2})$ to switch number $(j+m-1)$ *mod* $\frac{n^3}{2}$. Intuitively we are just shifting in a circular way the pattern of connection of nodes in the first cluster to the other clusters.

The interconnection is represented as a $\frac{n^3}{2} \times n$ matrix L (see Figure 5) such that $L(i,j)$ $(\forall i \in \{1..\frac{n^3}{2}\}$ and $\forall j \in \{1..n\})$ is

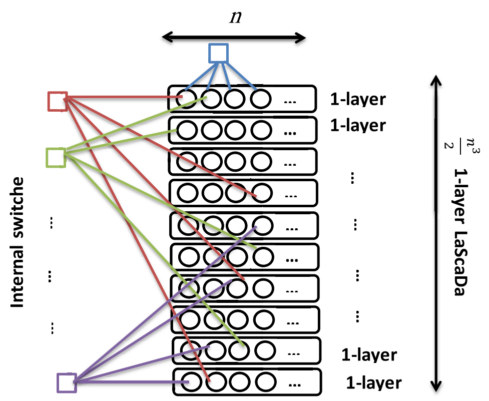


Fig. 5. 2-layer LaScaDa topology.

the index of the internal switch to witch node (i, j) (i.e., node j in cluster i) is connected to.

To generate matrix L we need first to generate its first row $L(1)$ we denote L_1 to distinguish it from the other rows. Remaining rows are systematically derived as follows:

$$\forall i \in \{2..\frac{n^3}{2}\}, \forall j \in \{1..n\} \quad L(i,j) = (L(i-1,j)+1)\ mod\ \frac{n^3}{2}.$$

In words, this means that each remaining row is derived from the row before it by simply adding one to each entry modulo the number of rows in matrix L. By doing so, we replicate the connection pattern of the first cluster to the remaining clusters by progressively shifting it by one, while wrapping it back on the list of clusters when the end of this list is reached. This allows also to propagate the characteristics of the first cluster to the other clusters. For instance, if the number of clusters directly connected to the first cluster is maximized, then this number will also be maximized for remaining clusters.

To generate L_1, we propose a novel algorithm we call *Linked Clusters Maximization* (LCM) (see Algorithm 1). This algorithm maximizes the number of directly connected clusters for each cluster, which leads to a reduction in the number of intermediate hops needed to transmit a packet to its destination (i.e., reduces the APL).

In Algorithm 1, the first element of vector L_1 is initialized to 1, then $\forall i \in \{2..n\}$, the best internal switch to be connected to node $(1, i)$ is selected by computing the size of the linked clusters set for each possible switch j ($j = 1..\frac{n^3}{2}$). The internal switch S^* maximizing the number of connected clusters is selected for node $(1, i)$ by setting $L_1(i) = S^*$, which corresponds to setting $L(1, i) = S^*$ in matrix L. Note that when using n-port switches, the maximum number of clusters a cluster can be directly connected to is equal to $n(n-1)$. This is due to the fact that a cluster has n nodes, each one of which can be connected to one distinct internal switch, which in turn can connect to $(n-1)$ different clusters. In Algorithm (1) we adopt a greedy approach to find the best internal switch. An exact approach would be computationally very expensive, even for small values of n, as we will have to check all possible configurations of Matrix L. In our approach, there is no need to check all possible internal switches $\forall j \in \{1..\frac{n^3}{2}\}$ all the time to find the best one

Algorithm 1 Linked Clusters Maximization

```
procedure $LCM(n)$
    $j_{selected}$ is the index of the selected internal switch.
    $D$ is a connectivity vector of size $n$.
    Input:
    $n$ is the number of column of matrix $L$.
    Output:
    $L_1$ is the first row of the matrix $L$.
    $\Omega$ is the set of linked clusters distances.

    $L_1(1) \leftarrow 1$, $j_{selected} \leftarrow 0$;
    for $i \leftarrow 2\ to\ n$ do
        $D(1..n) \leftarrow 0 // set\ all\ entries\ of\ D\ to\ 0$
        for $j \leftarrow 1\ to\ \frac{n^3}{2}$ do
            $L_1(i) \leftarrow L_1(i-1) + j$
            $D(j) \leftarrow |LinkedClusters(i, L_1)|$
            if $D(j) = i \times (i-1)$ then
                Break
            end if
        end for
        $j_{selected} \leftarrow\ index\ of\ the\ max\ value\ in\ D$
        $L_1(i) \leftarrow L_1(i-1) + j_{selected}$
    end for
    $\Omega = \{(x-1)\ for\ x\ in\ LinkedClusters(n, L_1)\}$
end procedure
function $LinkedClusters(p, L_1)$
    Input:
    $L_1$ is the first line of the matrix $L$.
    $p$ is the internal switch index
    Output:
    $LC$ is the connected clusters set for the first cluster.

    $LC \leftarrow \emptyset$
    for $i \leftarrow\ 1\ to\ p$ do
        for $j \leftarrow\ 1\ to\ p$ do
            if $L_1(i) \neq L_1(j)$ then
                $Add\ ((L_1(i) - L_1(j))\ mod\ \frac{n^3}{2}) + 1\ to\ LC$
            end if
        end for
    end for
    return $LC$;
end function
```

to connect to. At step i ($\forall i \in \{2..n\}$), we consider that every switch has only i ports available. If an internal switch that connects a number of clusters equal to the maximum number of clusters using i-port switches is found (i.e., $i(i-1)$), then it is directly selected without the need to check further internal switches. Otherwise, the next best switch is selected.

After L_1 is generated, the set Ω of linked clusters distances is computed as stated in Algorithm 1. The set Ω is such that $\forall i \in \{1..\frac{n^3}{2}\}$ and $\forall j \in \Omega$, cluster i and cluster $(i+j)\ mod\ \frac{n^3}{2}$ are directly connected. Figure 6 shows the network topology of LaScaDa built using 4-port switches. To connect 128 nodes based on LaScaDa topology, 32 internal and 32 external 4-port switches are used. For clarity reasons,

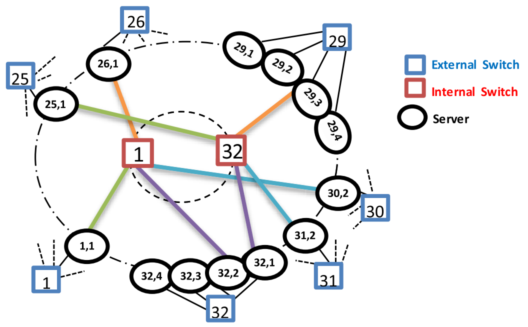


Fig. 6. LaScaDa network for n=4 and k=2.

only connections to internal switches 1 and 32 and few external switches are shown in Figure 6.

For a k-layer LaScaDa network with $k > 2$, the total number of connected nodes is $n(\frac{n^3}{2})^{k-1}$. Its connection pattern follows the same pattern as a 2-layer LaScaDa network. In fact, a 2-layer LaScaDa connects $\frac{n^3}{2}$ 1-layer LaScaDa following the pattern computed in matrix L. Similarly, a 3-layer LaScaDa connects $\frac{n^3}{2}$ 2-layer LaScaDa following the same pattern in matrix L. In general, a k-layer LaScaDa connects $\frac{n^3}{2}$ $(k-1)$-layer LaScaDa following the pattern in matrix L. Following this recursive structure, the label of a node in the k-layer LaScaDa network, representing also its coordinates in the topology, is built from its label in the $(k-1)$-layer LaScaDa network where it appears, prefixed with the index of that $(k-1)$-layer LaScaDa network. So, a node labeled $(C_{k-1},..,C_1)$ in the $(k-1)$-layer LaScaDa network number C_k, will be relabeled $(C_k, C_{k-1},..,C_1)$.

Algorithm 2 shows how to construct a k-layer LaScaDa network using $\frac{n^3}{2}$ $(k-1)$-layer LaScaDa networks numbered from 1 to $\frac{n^3}{2}$.

Figure 7 shows an example of LaScaDa network where $n = 2$, $k = 3$. The network is divided into four 2-layer LaScaDa connected using 16 intermediate switches.

Algorithm 2 Layered Linked Clusters Maximization

```
procedure LAYERED LASCADA($L$)
    $k$ is the network degree.
    Input:
    $L$ is the connection matrix.
    $\frac{n^3}{2}$ $(k-1)$-layer LaScaDa networks numbered from 1 to $\frac{n^3}{2}$
    Output:
    $k$-layer LaScaDa network

    /* Connect nodes/*
    for each tuple $(C_k, C_{k-1},..,C_i,..,C_1) \in \{1,..,\frac{n^3}{2}\}^{k-1} \times \{1,..,n\}$ do
        Connect node $(C_{k-1},..,C_i,..,C_1)$ in the $C_{k-1}$-layer LaScaDa network number $C_k$ to the internal switch number $(C_{k-1},..,C_2,L(C_k,C_1))$.
    end for
    /* Relabel nodes/*
    for each tuple $(C_k, C_{k-1},..,C_i,..,C_1) \in \{1,..,\frac{n^3}{2}\}^{k-1} \times \{1,..,n\}$ do
        Change the label of node $(C_{k-1},..,C_i,..,C_1)$ in the $C_{k-1}$-layer LaScaDa network number $C_k$ to $(C_k, C_{k-1},..,C_i,..,C_1)$.
    end for
end procedure
```

D. Illustrating the Execution of LCM Algorithm

In this section, an example of LaScaDa (k=2, n=4) is presented. LCM algorithm has been applied to determine the network connection pattern. The size of the connection matrix L is (32×4), meaning that the network is composed of 32 clusters and contains 128 nodes interconnected by means of 64 4-port switches (32 internal and 32 external switches).

Each row of matrix L (see Figure 8) corresponds to one cluster. The column index in L corresponds to the index of the node in the cluster, and the numbers in the matrix (Figure 8) refer to the index of the connected internal switch. The first row L_1 of L is generated using LCM, while each remaining row is systematically deduced from the row that immediately precedes it by adding 1 modulo the number of rows. As shown in Figure 8 (a), the first step of LCM algorithm consists in initializing the first column of the first row of L_1 to internal switch number 1. Remaining columns are set iteratively with the objective to maximize the number of clusters connected to the first cluster.[2] For each remaining column i of L_1, we check which internal switch provides the maximum number of connected clusters to cluster number 1. Theoretically, when setting the i^{th} column of L_1, and by assuming that each switch has only i ports, each cluster can be directly connected to a maximum to $i(i-1)$ clusters. Hence, the size of the connected clusters set LC is always compared to $i(i-1)$ to make sure that the switch choice is optimal.

For the column element of L_1 shown in Figure 8 (a), given that internal switch number 1 is already connected to cluster 1, the next available internal switch is 2. Hence, the second column of the L_1 is set to 2 to see if internal switch 2 is a good choice. As it is shown in Figure 8 (b), cluster number 1 becomes directly connected to cluster 2 (by means of internal switch 2) and cluster 32 (by means of internal switch 1) after the systematic completion of matrix L. The size of LC is 2, which is the maximum number of directly connected clusters that can be reached (i.e., $2 \times (2-1)$). So, LCM selects switch number 2 to be connected to the second node of the first cluster. For the third column, the theoretical maximum number of directly connected clusters is equal to 6 (i.e., $3 \times (3-1)$). LCM algorithm tries with the first available switch which is switch number 3. In this case, 4 clusters will be directly connected to cluster number 1 (Figure 8 (c)), which is less than the maximum. Therefore, switch number 3 will not be selected and LCM tries with switch number 4. In this case, clusters 2, 3, 4, 30, 31, and 32 are directly connected to cluster 1 and the size of LC becomes equal to the maximum

[2]Note that since the connection pattern is repeating, maximizing the number of clusters connected to the first cluster induces maximizing the number of connected clusters in the network

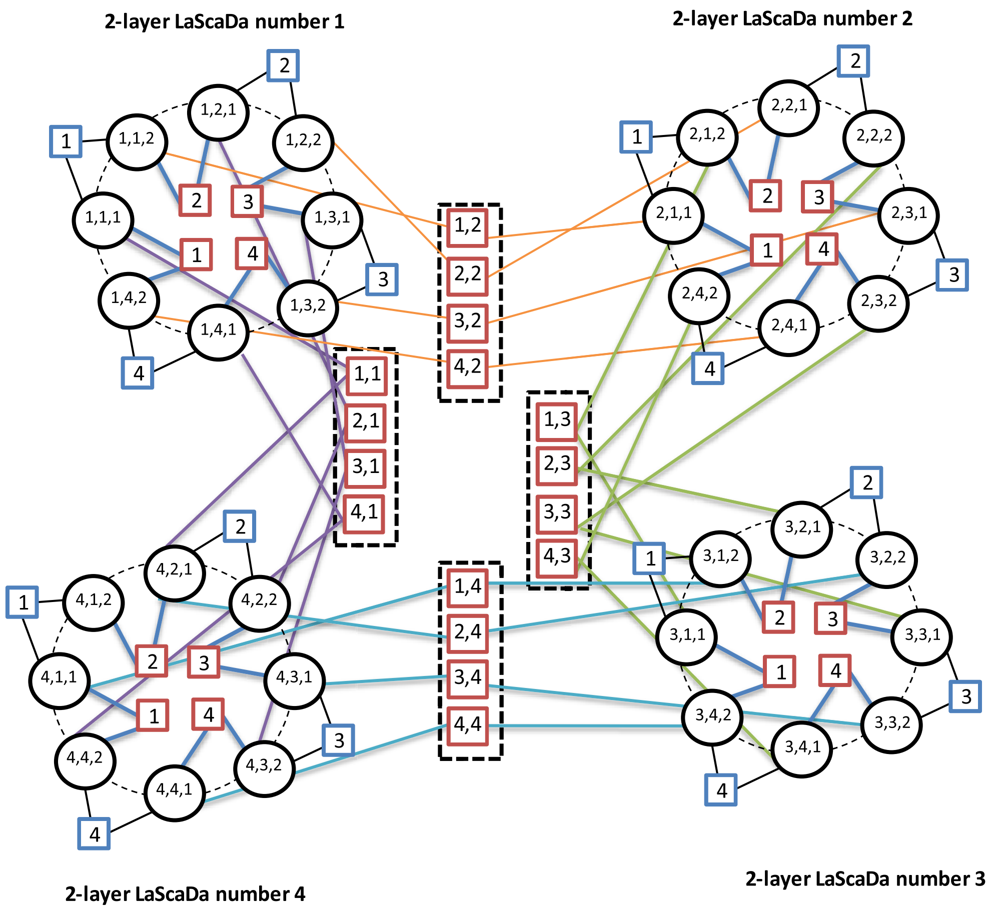


Fig. 7. LaScaDa network for n=2 and k=3.

(Figure 8 (d)). So, switch number 4 is selected to be connected to node number 3 of the first cluster. The same process is repeated for the fourth column for which switch number 8 is selected because it achieves the maximum number of directly connected clusters which is equal to 12, which can be seen in Figure 8 (e).

The time complexity of algorithm LCM for computing matrix L is $O(n^4)$, where n is the number of ports per switch. Given that Matrix L interconnects $\frac{n^3}{2} \times n$ nodes, the complexity of LCM is linear in terms of the number of nodes. Knowing also that n is in general small, matrix L can be computed in a relatively short time. It is also computed once but has multiple uses, including deriving the topology of a multilayer LaScaDa and routing parquets as it will be explained later on.

IV. LaScaDa Key Features

In this section, we analyze the key features of LaScaDa and compare them with features of other topologies. Some quantitative structural properties of LaScaDa topology are presented in Table III and Table V where n is the number of ports per switch and k is the number of ports per node.

A. Diameter

Given the shortest distances between all pairs of nodes, the diameter is defined as the maximum of these distances. Although LaScaDa connects a greater number of nodes compared with all other topologies, it has the lowest diameter when compared with Ficonn and Flecube (See Figure 9). For a large value of k, the diameter of LaScaDa is approximately equal to the diameter of HyperBcube and BCube. Thanks to its routing algorithm, LaScaDa reduces the APL even for a big number of nodes.

B. Scalability and Physical Cost

The topology of LaScaDa shows great scalability in terms of the number of nodes. Table III presents the number of nodes under different configurations. Table IV, on the other hand, shows an example of a configuration with $n = 48$, $k = 2$ and $n_{ToR} = 100$, $N_{sw} = 3200$, $r = 36$. In this example, a significantly larger data center network can be constructed with LaScaDa topology. However, the number of links and switches per node in LaScaDa is larger than the number of links and switches per node in Clos Network, and less than the number of links and switches per node in FatTree and Jellyfish. Basically, the performance of Jellyfish and Clos Network is highly influenced by their configuration parameters which are n_{ToR}, N_{sw}, and r.

Figure 11 shows the number of nodes in LaScaDa compared to Clos Network, FatTree, Jellyfish and VL2. The number of ports per switch is varied from 4 to 64. n_{ToR} is varied between 100, 200, 300, and r is varied between $\frac{n}{4}, \frac{n}{2}, \frac{3n}{4}$. We can see

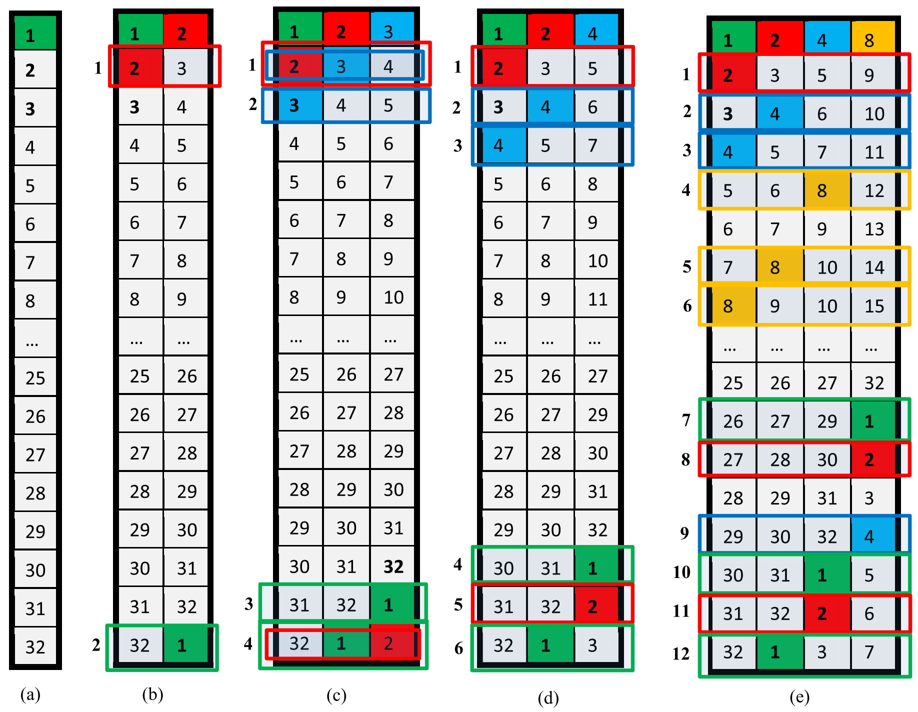


Fig. 8. Connections Matrix computation using LCM algorithm when $k = 2,\ n = 4$.

TABLE III
COST COMPARISON WITH TOPOLOGIES SUPPORTING 2 LAYERS ONLY

	Vl2	FatTree	Clos Network	Jellyfish	LaScaDa
Nodes Number	$\frac{(n-2)n^2}{4}$	$\frac{n^3}{4}$	$\frac{n^2}{4} \times n_{ToR}$	$N(n-r)$	$\frac{n^4}{2}$
Link Number	$\frac{(n+2)n^2}{4}$	$3\frac{n^3}{4}$	$n^2(1+\frac{n_{ToR}}{4})$	$N \times n$	n^4
Switches Number	$\frac{3n}{2} + \frac{n^2}{4}$	$5\frac{n^2}{4}$	$\frac{3n}{2} + \frac{n^2}{4}$	N	n^3
Network Diameter	6	6	6	4	5

TABLE IV
AN EXAMPLE OF CONFIGURATION WITH $n = 48$, $k = 2$ AND $n_{TOR} = 100$ $N_{sw} = 3200, r = 36$

	Vl2	FatTree	Jellyfish	Clos Network	LaScaDa
Nodes Number	26496	27648	38400	57600	2×10^6
Switches Number	648	5760	3200	648	110592
Link Number	28800	82944	153600	59504	2×10^6
Link Number Per Node	1.08	3	4	1.04	2
Switches Number Per Node	0.24	0.1	0.08	0.01	0.04
Network Diameter	6	6	6	4	5

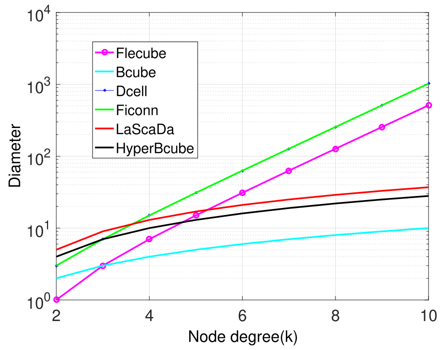


Fig. 9. Network diameter of various layered topologies.

that the number of nodes in Clos topology increases with n_{ToR}, however, Jellyfish connects a small number of nodes when r increases. LaScaDa, on the other hand, connects a larger number of nodes compared to the other topologies.

Figure 12 shows the number of links per device (server or switch) in LaScaDa compared to Clos Network, FatTree, Jellyfish and VL2. The number of ports per switch is varied from 4 to 64, n_{ToR} is varied between 100, 200, 300, and r is varied between $\frac{n}{4}, \frac{n}{2}, \frac{3n}{4}$. The number of links per node in Jellyfish varies with r and is equal to 4 when $r = \frac{3r}{4}$. In particular, the larger r, the larger is the number of links per node. On the other hand, LaScaDa still maintains a modest average cost in terms of links per server (2 links per node). The number of links per server in LaScaDa is identical to a tiny 2-layer BCube, and less than Dcell (Figure 10). Moreover, and thanks to its repeated connection pattern (Section C), LaScaDa maintains an acceptable wiring/cabling complexity (e.g., pre-manufactured standardized connection modules connect the servers in a column through one single action).

Table V shows a cost comparison for layered topologies. With only 4-port or 6-port switches and 4 to 6 recursive layers,

TABLE V
COST COMPARISON WITH LAYERED TOPOLOGIES

	HyperBcube	DCell	BCube	LaScaDa
Nodes Number	n^{2k-1}	$a_1 = n(k=1)$ $a_k = a_{k-1}(a_{k-1}+1)(k \geq 2)$	n^k	$\frac{n^{3k-2}}{2^{k-1}}$
Node degree	k	k	k	k
Link Number	kn^{2k-1}	$(k+1)\frac{a_k}{2}$	kn^k	$\frac{kn^{3k-2}}{2^{k-1}}$
Switches Number	kn^{2k-2}	$\frac{a_k}{n}$	kn^{k-1}	$\frac{kn^{3(k-1)}}{2^{k-1}}$
Connection Pattern	Partially connected	Fully connected	Fully connected	Partially connected

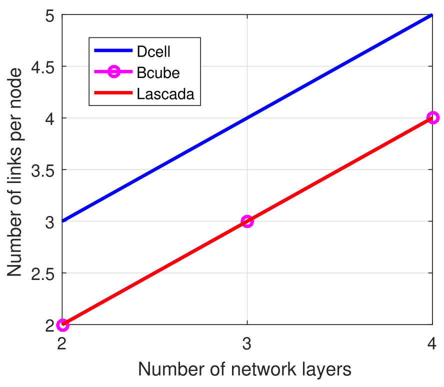


Fig. 10. The number of nodes in LaScaDa compared to BCube and Dcell.

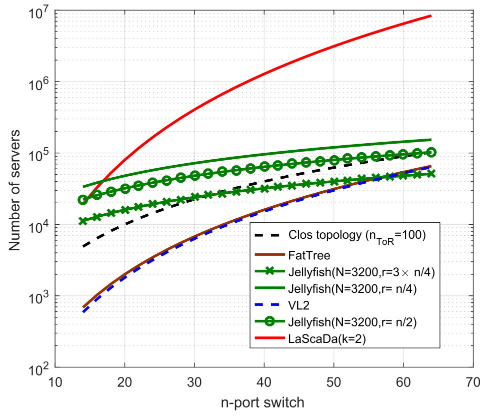


Fig. 11. The number of nodes in in LaScaDa compared to Clos Network, FatTree, Jellyfish and VL2.

the number of nodes overpasses the million (7×10^6). Using identical n-port switches and only a two-layer network, we can host $\frac{n^4}{2}$ nodes, which is approximately 5 times larger than what VL2 can host, and n^2 times larger than what DCell and BCube can host. Furthermore, with its excellent scalability, LaScaDa maintains a low average cost. In fact, the cost per node of LaScaDa is identical to the cost of a 2-layer BCube.

C. Bandwidth

The bandwidth is used to characterize data transfer rate, i.e., the amount of data that can be transferred from one point to another. There are four types of data bandwidths that can occur under different traffic patterns:

- One-to-One bandwidth: Represents the maximum bandwidth that the topology offers when one arbitrary node sends data to another arbitrary node.

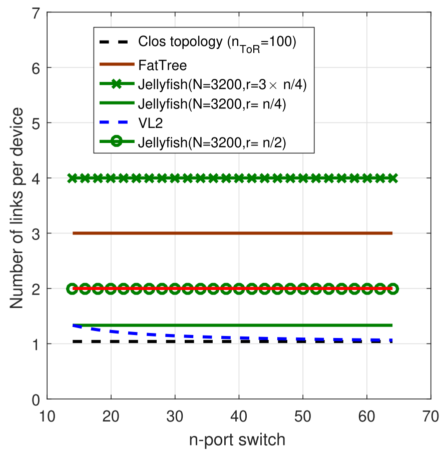


Fig. 12. The number of links per device in LaScaDa compared to Clos Network, FatTree, Jellyfish and VL2.

- One-to-All bandwidth: Occurs when updating some software on all nodes.
- One-to-Several bandwidth: Occurs when the file system is making replicas.

The One-to-One, One-to-Several, and One-to-All bandwidths are limited by the number of ports on each node (nodes degree k). So, for a tree-based topology, the bandwidth equals 1, while for a recursive topology the bandwidth k. Consequently, the basic tree topology has the smallest All-to-All bandwidth because of the limited number of switch ports at the root. In addition, this indicates that LaScaDa offers a great bandwidth performance under any traffic configuration ($k \geq 2$).

D. Bisection Bandwidth

The Bisection Bandwidth is used to measure the worst-case network capacity. Networks with higher bisection bandwidth are more resilient to link failures [3] because increasing the bisection bandwidth creates path diversity around points of failure. Many proposed topologies deploy various configurations to improve the bisection bandwidth. For example, Facebook's current data center network uses multiples racks to introduce path variety. Each rack contains a rack switch (RSW) with up to forty-four 10G downlinks and four or eight 10G uplinks (typically 10:1 oversubscription), one to each cluster switch (CSW). A cluster is a group of four CSWs with corresponding server racks and RSWs. Each CSW has four 40G uplinks (10G-4), one to each of four aggregation switches (typically 4:1 oversubscription).

TABLE VI
THE PERFORMANCE UNDER DIFFERENT CONFIGURATIONS (FAULT-FREE)

BCube				DCell				HyperBcube				LaScaDa			
Nodes	n	k	APL	Nodes	n	k	APL	Nodes	n	k	APL	Nodes	n	k	APL
100	10	2	3.55	110	10	2	4.25	100	5	2	5.4	128	4	2	3.55
625	25	2	3.75	600	24	2	4.7	512	8	2	6	648	6	2	4.13
1225	35	2	4	1260	35	2	4.8	1331	11	2	6.1	2048	8	2	4.41
4096	4	6	4.5	1806	6	3	5.73	4096	16	2	6.34	32768	16	2	4.51
16384	4	7	5.25	176820	4	4	11.29	13824	24	2	6.36	80000	20	2	4.61

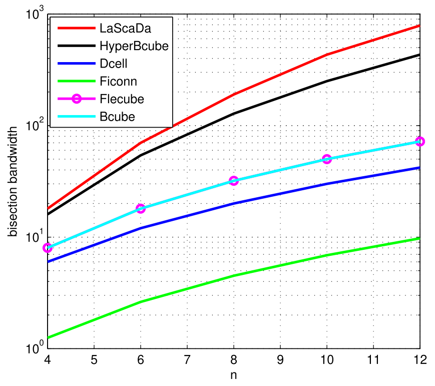


Fig. 13. Bisection bandwidth of various layered topologies.

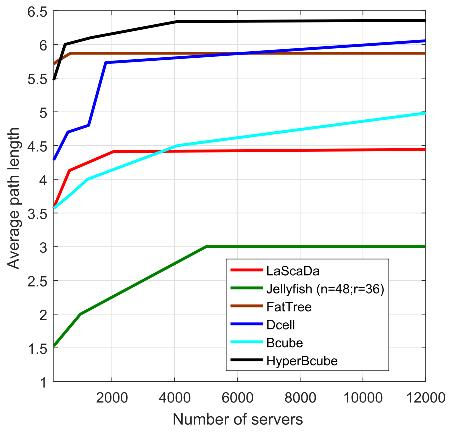


Fig. 14. The APL comparison of LaScaDa compared to FatTree, Jellyfish, DCell and BCube.

Fat-tree also can guarantee a 1:1 over-subscription to support non-blocking communications between servers and significantly improve the performance of data center networks. In order to avoid the oversubscription and increase the bisection bandwidth, LaScaDa increases the number of connected clusters and the number of links per node. As stated before, a LaScaDa network can be represented as a $\frac{n^3}{2} \times \frac{n^{(3k-5)}}{2^{(k-2)}}$ matrix. Alternatively, it can be seen as $(\frac{n^3}{2})^{(k-2)}$ 2-layer LaScaDa network, each one with $\frac{n^3}{2} \times n$ nodes. Let B_G denotes the bisection bandwidth of a 2-layer LaScaDa network. Given that the connection pattern of nodes repeats $(\frac{n^3}{2})^{(k-2)}$ times, we deduce that the total bisection bandwidth is equal to $B_G \times (\frac{n^3}{2})^{(k-2)}$. However, an exact expression for B_G is hard to obtain since this depends on the output of LCM algorithm for each switch port. In theory, the number of connected clusters could be maximized using the LCM algorithm. However, the exact number cannot be known. Hence, the layout of the topology can slightly change from one port to another.

Experimental results comparing LaScaDa to other topologies are shown in Figure 13. LaScaDa can substantially increase the network capacity as it has the largest value of bisection bandwidth compared to other topologies. For Instance, with 12-port switches and a node degree of 2, the bisection bandwidth of a 10368-node data center is 63%, 994%, 1776% and 7780% bigger than the bisection bandwidth of HyperBcube, BCube, DCell, and Ficonn respectively.

E. Average Path Length

The Average Path Length (APL) is used to evaluate the overall performance of the whole network due to its impact on packet delays [26].

According to Table III, Jellyfish has the lowest diameter, which is clearly shown by its APL in Figure14. Jellyfish has the lowest APL compared to the other topologies, meaning that it is able to connect larger number of nodes with smaller APL. LaScaDa also reduces the APL compared to FatTree, DCell and BCube.

Table VI shows that LaScaDa provides a higher scalability and a much smaller APL than the other topologies, even with a small node degree ($k = 2$). LaScaDa reduces dramatically the APL compared to other topologies. For instance, for 32768 servers, the APL of LaScaDa is 4.51, the APL of HyperBCube is 6.36 for 13824 nodes, the APL of BCube is 5.25 for 16384 nodes and it is 5.73 for 1806 nodes DCell. This means that LaScaDa reduces the APL by 29%, 14%, and 21$\%$ compared to HyperBCube BCube and DCell, respectively. In addition, we can observe that BCube, DCell, and HyperBcube require more than three network layers to scale up to a large size. In fact, LaScaDa connects 80×10^3 nodes with 20-port switches and two-layer network, while HyperBcube connects 13824 nodes with 24-port switches (the scalability increases by 80% using less n-port switches).

F. Aggregate Bottleneck Throughput

The Aggregate Bottleneck Throughput (ABT) indicates the sum of the throughput of all bottleneck flows [27]. LaScaDa achieves high ABT. For instance, the ABT of a 2048-node LaScaDa under All-to-All traffic patterns is 920. The total number of two-way communication links is $N_{Links} = 4096$. The proportion of the overall network capacity that ABT can reach is $NCP_{ABT} = ABT/2N_{links} = 11.23\%$, which is already very close to its theoretical limit ($NCP_{ABT} = 1/APL$ = 1/8.8 = 11.36%). This reveals the great performance of LaScaDa in terms of Aggregate Bottleneck Throughput.

G. Incremental Expansion

The proposed topology uses an incremental construction approach. In fact, the LaScaDa network has a repeating structure that can be built partially. So, even if an entire column is missing from matrix L, the system performance will not be reduced. For instance, if $\frac{1}{n}$ of columns are missing, the entire network can still be represented as a complete LaScaDa network built using $(n-1)$-port switches.

V. Routing Scheme

A. Fault Free Routing Scheme

To forward a packet from a source $(S_k, S_{k-1}, \dots, S_1)$ to a destination $(D_k, D_{k-1}, \dots, D_1)$, we propose a hierarchical row-based routing algorithm. A path P can be established using the following k steps, where only one coordinate is used in each step:

$P = (S_k, S_{k-1}, \dots, S_1) \rightarrow (D_k, ?, \dots, ?) \rightarrow (D_k, D_{k-1}, \dots, ?) \rightarrow \dots \rightarrow (D_k, D_{k-1}, \dots, D_2, ?) \rightarrow (D_k, D_{k-1}, \dots, D_2, D_1)$

where "?" denotes unknown/don't care value.

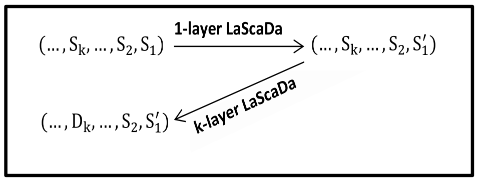
For $k = 2$, to forward packets from node (S_2, S_1) to node (D_2, D_1), we propose a cluster-based fault free routing scheme as shown in Algorithm 3. If the source and destination have the same second coordinate, they are in the same cluster and therefore are directly connected via an external switch. However, if the modular difference between the source and destination cluster belong to the set of linked clusters distances Ω (i.e., $(D_2 - S_2)\ mod\ \frac{n^3}{2}) \in \Omega$), then an internal switch can be used to connect these nodes with a maximum of 3 hops. If the modular difference does not belong to Ω, then 2 or 3 internal switches have to be used to forward the packet with up to 5 hops.

Given that the rows in the interconnection matrix L are derived by progressively incrementing its first row L_1 by one modulo the number of rows (i.e., $\frac{n^3}{2}$), the route used to forward a packet from (S_2, S_1) to (D_2, D_1) can be directly deduced from the route used to forward the packet from $(1, S_1)$ to $((D_2 - S_2)\ mod\ \frac{n^3}{2}, D_1)$ by adding D_2 to the second coordinate of each node in the route. For instance, if $(1, S_1)$ is connected to (D_2, D_1) via the intermediate nodes (T_2^1, T_1^1) and (T_2^2, T_1^2), i.e., via the path $(1, S_1) \rightarrow (T_2^1, T_1^1) \rightarrow (T_2^2, T_1^2) \rightarrow (D_2, D_1)$, then (S_2, S_1) is connected to $((D_2 - S_2)\ mod\ \frac{n^3}{2}, D_1)$ via the path $(d, S_1) \rightarrow ((T_2^1 + d)\ mod\ \frac{n^3}{2}, T_1^1) \rightarrow ((T_2^2 + d)\ mod\ \frac{n^3}{2}, T_1^2) \rightarrow ((D_2 + d)\ mod\ \frac{n^3}{2}, D_1)$, where $d = (D_2 - S_2)\ mod\ , \frac{n^3}{2}$. Algorithm 3 constructs the route from the nodes of the first cluster to the other nodes, while the other routes can be directly deduced.

For $k > 2$, Algorithm 3 can be generalized for a multi-layer LaScaDa with two different cases as shown in Figure 15 and Figure 16. In case (a) (see Figure 15) when $(S_k - D_k)\ mod\ \frac{n^3}{2} \in \Omega$, D_k and S_k are directly connected to a common switch. Thus, a path $(\dots, S_k, \dots) \rightarrow (\dots, D_k, \dots)$ exists. Given a random position in the source row, one additional transition through a 1-layer LaScaDa may be required, leading to a maximum path length of two in the worst case.

Algorithm 3 Fault Free Routing Algorithm

```
procedure $FaultFreeRouting((S_2, S_1), (D_2, D_1), \Omega)$
    Input:
    $\Omega$ is the vector of directly connected clusters
    $(S_2, S_1)$ is the source coordinates
    $(D_2, D_1)$ is the destination coordinates
    Output:
    $Path$ is the path from the source to the destination
    if $S_2 = D_2$ then
        /*The source and the destination are in the same
          cluster and are directly connected via an external
          switch/*
        $Path \leftarrow (S_2, S_1) \rightarrow (D_2, D_1)$
    else
        if $(S_2 - D_2)\ mod\ \frac{n^3}{2} \in \Omega$ then
            /*The source and the destination are directly
              connected/*
            Find $T_2^1$ and $T_1^1$ such that $P \leftarrow (S_2, S_1) \rightarrow
              (S_2, T_1^1) \rightarrow (D_2, T_2^1) \rightarrow (D_2, D_1)$.
        else if $(S_2 - D_2)\ mod\ \frac{n^3}{2} \notin \Omega$ then
            /*The source and the destination are not directly
              connected and are linked only by the interme-
              diate of 2 switches/*
            Find $T_1^1$, $T_2^1$, $T_1^2$ and $T_2^2$ such that $P \leftarrow
              (S_2, S_1) \rightarrow (T_2^i, T_1^i) \rightarrow (T_2^j, T_1^j) \rightarrow (D_2, D_1)$
              where $(i, j)$ is an arrangement of {1,2}
        else
            /*The source and the destination are not directly
              connected and are linked only by the interme-
              diate of 3 switches/*
            Find $T_1^1$, $T_2^1$, $T_1^2$, $T_2^2$, $T_1^3$ and $T_2^3$ such that
              $P \leftarrow (S_2, S_1) \rightarrow (T_2^i, T_1^i) \rightarrow (T_2^j, T_1^j) \rightarrow
              (T_2^k, T_1^k) \rightarrow (D_2, D_1)$ where $(i, j, k)$ is an
              arrangement of {1,2,3}.
        end if
    end if
    $Path \leftarrow P$
end procedure
```



$(\dots, S_k, \dots, S_2, S_1)$
1-layer LaScaDa
$(\dots, S_k, \dots, S_2, S_1')$
k-layer LaScaDa
$(\dots, D_k, \dots, S_2, S_1')$

Fig. 15. Case (a) when $(S_k - D_k)\ mod\ \frac{n^3}{2} \in \Omega$.

In case (b) (see Figure 16), when $(S_k - D_k)\ mod\ \frac{n^3}{2} \notin \Omega$, there is no direct connection between D_k and S_k, thus a path $(\dots, S_k, \dots) \rightarrow (\dots, I_k, \dots) \rightarrow (\dots, D_k, \dots)$ is taken, where I_k denotes a common intermediate row between D_k and S_k. Accordingly, the path length is increased, leading to a maximum length of 3 intermediate switches in the worst-case.

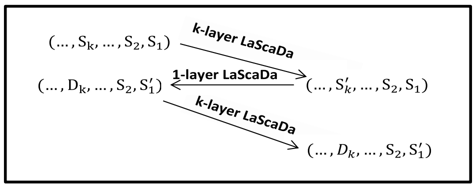


Fig. 16. Case (b) when $(S_k - D_k)\ mod\ \frac{n^3}{2} \notin \Omega$.

B. Fault-Tolerant Routing Scheme

The fault-tolerant routing is proposed for the fault-tolerant case in LaScaDa. Given a pair of nodes (.i.e., source S_k wants to communicate with destination D_k (unicast)), according to the values of $MaxlifeTime$, the algorithm tries to find alternative paths to forward packets to the destination. It starts by looking for a new source in the same cluster as S_k, otherwise, it looks in the neighborhood of the cluster of S_k (the vector of clusters connected to the cluster of S_k via exactly $k+1$ internal switches). We define the Connection Failure Rate (CFR) metric to measure the possibility that the routing protocol cannot find an available route. If CFR increases slowly with the growth of the number of faulty devices in the data center network, this indicates that the network can still maintain acceptable performance under faulty conditions. Results show that LaScaDa has good Fault-Tolerance. On the other hand, and because of the detours that should be taken to work around the failures, the average path length and the latency of LaScaDa will increase as the link failure rate increases. Hence, there will always be trade-offs between performance and reliability.

Algorithm 4 Fault Tolerant Routing Algorithm

```
1: procedure $FaultTolerantRouting(MaxlifeTime)$
2:     NHops is the used number of hops
3:     Intput:
4:     $MaxLifeTime$ is maximum number of hops
       ------------------------------------------
5:     NHops=0
6:     while Routing failed and NHops<MaxlifeTime do
7:         Find nearby severs in a radius of NHops and try
               routing by supposing the selected node as new
               source
8:         Select only Routes shorter than $MaxlifeTime$
9:         NHops=NHops+1
10:    end while
11: end procedure
```

VI. Experimental Results

Figure 17 shows the scalability of LaScaDa under different port switch and node degree configurations. The figure shows that by using a small port count switch n and high node degree k, the scalability of the topology increases much faster than when using a big n and a small k. As a result, a good tradeoff between cost and performance would be to use only small port count switches and a high node degree.

Figure 18 presents the average path length of LaScaDa under different configurations. Switches port count is varied from 4 to 8, while the node degree is varied from 2 to 6. First, it can be seen that the APL increases proportionally to the node degree k. So, a larger size LaScaDa has longer APL. However, we can see that by increasing the number of nodes, the APL takes values from 3.8 for 128 nodes (k=2, n=4), to 14.9 for 134×10^6 nodes (k=6, n=4). So, even if the number of nodes has increased by more than one million times, the APL did not exceed 15, and increased only by 3.9 times. In fact, thanks to its physical structure and routing algorithms, LaScaDa increases the number of directly connected clusters while reducing the number of intermediate hops during packet transmission. Hence, the APL has small values and does not reach the maximum value.

Figure 19 depicts the distribution of the number of nodes under different configurations. Switches port count is varied from 4 to 12, while node degree is equal to 2. First, we can see that LaScaDa and HyperBcube support larger numbers of nodes compared with all other topologies. Besides, results show that by increasing the switch degree n, the difference between LaScaDa and all other topologies greatly increases. For instance, the number of nodes increases by 133% for $n=12$. This shows the outperformance of LaScaDa in terms of scalability.

Figure 20 presents the connection failure rate as a function of the link failure rate of LaScaDa. $MaxLifeTime$ is varied between 6 and 5, the link failure rate is varied from 4% to 24%, while the number of nodes is fixed at 2048. For each failure rate value, we measure the connection failure rate as the proportion of failure to finding routes. If the connection failure rate increases slowly compared to the link failure rate, then this is a good indication that the network can maintain acceptable performances under faulty conditions.

From Figure 20, we can also see that $MaxLifeTime$ affects the connection failure rate of LaScaDa. In fact, when the link failure is 24% (around $\frac{1}{4}$ of links fail), the connection failure rate is only 10.41% for $MaxlifeTime$=6 hops, and 16.84% for $MaxlifeTime$=5 hops (equal to the diameter). This is due to the fact that the number of linked clusters in LaScaDa is large, which increases the number of alternative links to use in case of failures. Moreover, increasing the value of $MaxlifeTime$ helps in finding alternative paths to forward packets to the destination. Besides, for a small link failure rate, 6 hops are enough to find a route between any two nodes that are not totally disconnected.

VII. Discussion

Our preliminary investigation reveals that Lascada topology exhibits good properties that strike a good compromise between the double exponential scalability of DCell and the high-cost of BCube. In fact, Lascada scales data centers to large sizes without a noticeable loss in performance compared to existing topologies. Lascada can accommodate $\frac{n^4}{2}$ nodes (k=2), which is approximately 5 times larger than what Portland/VL2 can accommodate, and n^2 times larger than what DCell/BCube can accommodate. Besides its excellent scalability, LaScaDa maintains a lower-average cost compared

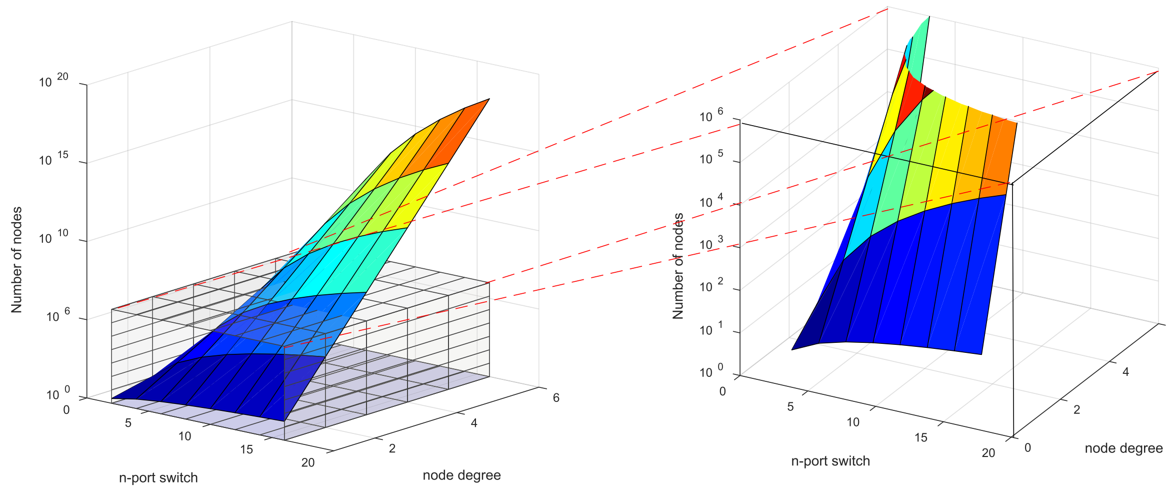


Fig. 17. Number of nodes under different port switch and node degree configurations.

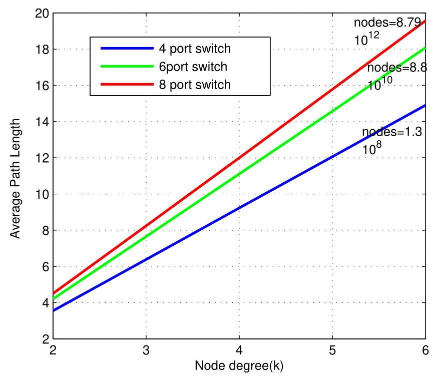


Fig. 18. Average path length of LaScaDa under different configurations.

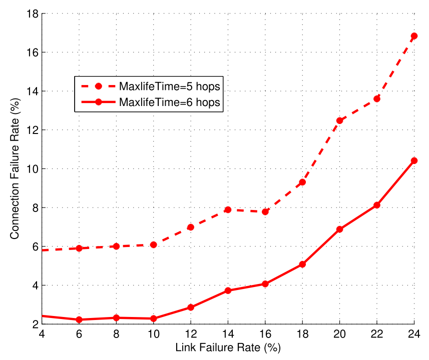


Fig. 20. The performance of a 2048-node LaScaDa with different values of $MaxLifeTime$.

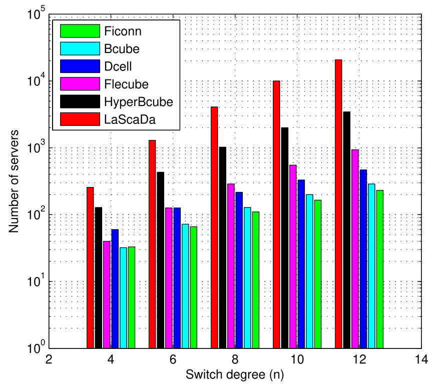


Fig. 19. The number of nodes of LaScaDa, FlatNet, Dcell and Bcube under different switch port count configurations.

to Bcube and Dcell. For instance, using a 6-port switch and 4 layers, Bcube connects only $6^4 = 1296$ nodes and requires 4 NICs per node, which is costly and hard to control in practice. Consequently, BCube has scalability issues for cost-effective small degree nodes and small-port-count switches. On the other hand, using 5-port switches and 3 layers, Dcell connects 1806, while LaScaDa connects 69984 (54 times greater). In terms of APL, Jellyfish has the lowest APL compared to the other topologies, meaning that it is able to connect a greater number of nodes with a shorter APL. However, the performance of Jellyfish and Clos Networks is highly influenced by their configuration parameters: n_{TOR}, N_{sw}, and r. Jellyfish's unstructured design brings also new challenges in routing, physical layout, and wiring. For LaScaDa, the APL is reduced when compared with FatTree, DCell, and BCube. In terms of Bandwidth, the basic tree topology has the smallest All-to-All bandwidth because of the limited number of switch ports at the root. Recursive topologies, including LaScaDa, offer a better bandwidth performance under any traffic configuration ($k \geq 2$).

VIII. Conclusions

In this paper, we proposed and evaluated a novel topology for data centers called LaScaDa. The proposed topology scales data centers to large sizes without a noticeable loss in performance compared to existing topologies. By using exclusively β links and small port count switches, LaScaDa scales up a data center to millions of nodes while preserving a good quality of service. LaScaDa is characterized by its high Scalability, high Aggregate Bottleneck Throughput, a good Fault-Tolerance, a low Average Path Length, and a high Bisection Bandwidth. Thanks to its special connections pattern and routing algorithms, LaScaDa tries to maximize the number of directly connected clusters. Simulation results confirm the efficiency and outperformance of our proposed topology.

Acknowledgment

This publication was made possible by NPRP Grant 6-718-2-298 from the Qatar National Research Fund (a member of Qatar Foundation), and by Qatar University Internal Grant No. IRCC-2020-001. Open Access funding provided by the Qatar National Library. The statements made herein are solely the responsibility of the authors.

References

[1] M. Chen, H. Jin, Y. Wen, and V. C. M. Leung, "Enabling technologies for future data center networking: A primer," *IEEE Netw.*, vol. 27, no. 4, pp. 8–15, Jul. 2013.

[2] P. Ruiu, A. Bianco, C. Fiandrino, P. Giaccone, and D. Kliazovich, "Power comparison of cloud data center architectures," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[3] E. Baccour, S. Foufou, and R. Hamila, "PTNet: A parameterizable data center network," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2016, pp. 1–6.

[4] S. Stroh, G. Schrder, and F. Grne. (2009). *Keeping the Data Center Competitive Six Levers for Boosting Performance, Reducing Costs, and Preparing for an On-Demand World*. [Online]. Available: http://www.strategyand.pwc.com/media/file/Keeping_Data_Center_Competitive.pdf

[5] Z. Chkirbene, A. Gouissem, R. Hadjidj, S. Foufou, and R. Hamila, "Efficient techniques for energy saving in data center networks," *Comput. Commun.*, vol. 129, pp. 111–124, Sep. 2018.

[6] R. N. Mysore *et al.*, "PortLand: A scalable fault-tolerant layer 2 data center network fabric," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 39–50, 2009.

[7] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, and S. Lu, "FiConn: Using backup port for server interconnection in data centers," in *Proc. IEEE INFOCOM-28th Conf. Comput. Commun.*, Apr. 2009, pp. 2276–2285.

[8] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," in *Proc. ACM SIGCOMM Conf. Data Commun. (SIGCOMM)*, 2008, pp. 75–86.

[9] C. Guo *et al.*, "BCube: A high performance, server-centric network architecture for modular data centers," in *Proc. ACM SIGCOMM Conf. Data Commun. (SIGCOMM)*, 2009, pp. 63–74.

[10] D. Lin, Y. Liu, M. Hamdi, and J. Muppala, "Hyper-BCube: A scalable data center network," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 2918–2923.

[11] Z. Chkirbene, S. Foufou, and R. Hamila, "VacoNet: Variable and connected architecture for data center networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2016, pp. 1–6.

[12] Z. Chkirbene, S. Foufou, M. Hamdi, and R. Hamila, "ScalNet: A novel network architecture for data centers," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2015, pp. 1–6.

[13] Z. Chkirbene, S. Foufou, R. Hamila, Z. Tari, and A. Y. Zomaya, "LaCoDa: Layered connected topology for massive data centers," *J. Netw. Comput. Appl.*, vol. 83, pp. 169–180, Apr. 2017.

[14] A. Greenberg *et al.*, "Vl2: A scalable and flexible data center network," *SIGCOMM Comput. Commun*, vol. 4, pp. 51–62, Aug. 2015.

[15] Y. Liu, J. Muppla, M. Veeraghavan, D. Lin, and M. Hamdi. (2013). *Data Center Network*. [Online]. Available: http://www.amazon.ca/Data-center-Network

[16] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM Conf. Data Commun. (SIGCOMM)*, 2008, pp. 63–74.

[17] C. Kachris and I. Tomkos, "A survey on optical interconnects for data centers," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1021–1036, 4th Quart., 2012.

[18] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *Proc. 9th USENIX Conf. Netw. Syst. Design Implement.* Berkeley, CA, USA: USENIX Association, 2012, p. 17.

[19] X. Ye, Y. Yin, S. J. B. Yoo, P. Mejia, R. Proietti, and V. Akella, "DOS: A scalable optical switch for datacenters," in *Proc. 6th ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, Oct. 2010, pp. 63–74.

[20] K. Xia *et al.*, "Petabit optical switch for data center networks," Polytech. Inst., New York Univ., New York, NY, USA, Tech. Rep., 2010.

[21] B. Lebiednik, A. Mangal, and N. Tiwari, "A survey and evaluation of data center network topologies," 2016, *arXiv:1605.01701*. [Online]. Available: https://arxiv.org/abs/1605.01701

[22] Y. Yu and C. Qian, "FTDC: A fault-tolerant server-centric data center network," in *Proc. IEEE/ACM 24th Int. Symp. Qual. Service (IWQoS)*, Jun. 2016, pp. 1–2.

[23] D. Li, Y. Shen, and K. Li, "Flecube," *Comput. Commun.*, vol. 77, pp. 62–71, Mar. 2016.

[24] B. Wang, Z. Qi, R. Ma, H. Guan, and A. V. Vasilakos, "A survey on data center networking for cloud computing," *Comput. Netw.*, vol. 91, pp. 528–547, Nov. 2015.

[25] E. Baccour, S. Foufou, R. Hamila, and M. Hamdi, "WFlatnet: Introducing wireless in flatnet data center network," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2015, pp. 1–6.

[26] Z. Chkirbene, S. Foufou, M. Hamdi, and R. Hamila, "Hyper-flatnet: A novel network architecture for data centers," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, Jun. 2015, pp. 1877–1882.

[27] T. Wang, Z. Su, Y. Xia, Y. Liu, J. Muppala, and M. Hamdi, "SprintNet: A high performance server-centric network architecture for data centers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 4005–4010.

Zina Chkirbene received the bachelor's degree in computer science networks and telecommunications from the National Institute of Applied Science and Technology in 2011, the master's degree in electronic systems and communication networks from Polytechnic School in 2012, and the Ph.D. degree in computer science from the University of Burgundy, Dijon, France, in 2017. In 2012, she interned as a Research Assistant at Qatar University. Her research interests include data center networks, cloud computing, green computing, and software-defined network, as well as distributed and cloud computing systems.

Rachid Hadjidj received the Ph.D. degree in computer engineering from the University of Montreal (Ecole Polytechnique), CANADA, in 2006. He worked as a Research Associate in cybersecurity with Concordia University, Canada, an Assistant Professor with Qatar University, Qatar, and a Consultant with the Canadian National Defense, Fujitsu, and Bell Canada, Canada. He is currently working as an Assistant Professor with the Ahmed Bin Mohammed Military College (ABMMC), Qatar. His research interests include real-time systems modeling and verification, machine learning, cybersecurity, and digital forensics.

Sebti Foufou (Member, IEEE) received the Ph.D. degree in computer science from the University Claude Bernard Lyon I, France, in 1997, for a dissertation on parametric surfaces intersections. He has been with the Computer Science Department, University of Burgundy, France, since 1998. In 2005 and 2006, he worked as a Guest Researcher with the National Institute of Standards and Technology, Gaithersburg, MD, USA. He was with the Department of Computer Science and Engineering, Qatar University, from 2009 to 2017, where he has served as the Head of the Department for three academic years from 2012 to 2015. He joined NYUAD in September 2017. His research interests include geometric modeling for shape representations and image processing for face recognition. He is also interested in data models, data representation, and processing for product lifecycle management and smart machining systems.

Ridha Hamila (Senior Member, IEEE) held various research and teaching positions at TUT within the Department of Information Technology, Finland, from 1994 to 2002. From 2002 to 2003, he was a System Specialist with the Nokia Research Center and Nokia Networks, Helsinki. From 2004 to 2009, he was with the Etisalat University College, Emirates Telecommunications Corporation, United Arab Emirates. He is currently a Professor at the Department of Electrical Engineering, Qatar University, Qatar. His current research interests include mobile and broadband wireless communication systems, DSP algorithms for flexible radio transceivers, cellular, and satellites-based positioning technologies. In these areas, he has published more than 90 journal and conference papers most of them in the peered reviewed IEEE publications, filed two patents, and wrote numerous confidential industrial research reports. He has been involved in several past and current industrial projects, QNRF, Ooredoo, Finnish Academy projects, EU research, and education programs. He has supervised a large number of bachelor's/master's students and post-doctoral fellows. He has organized many international workshops and conferences.