IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 24 September 2023, accepted 9 October 2023, date of publication 17 October 2023, date of current version 27 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3325321

RESEARCH ARTICLE

Advancing Data Center Networks: A Focus on Energy and Cost Efficiency

ZINA CHKIRBENE[1], RIDHA HAMILA[1], (Senior Member, IEEE), ARAFAT AL-DWEIK[2], (Senior Member, IEEE), AND TAMER KHATTAB[1], (Senior Member, IEEE)
[1]College of Engineering, Qatar University, Doha, Qatar
[2]Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, United Arab Emirates
Corresponding author: Zina Chkirbene (zina.chk@qu.edu.qa)

This work was supported by Qatar University Internal Grant IRCC-2023-237. Open Access funding provided by the Qatar National Library.

ABSTRACT Data centers serve as the backbone for cloud computing, enterprise services, and infrastructure-based offerings. One area of ongoing research in data center networking focuses on innovating new topologies for large-scale node connectivity. These topologies must incorporate fault-tolerant and efficient routing algorithms. Consequently, the data center network topology must dynamically adapt to ever-changing application requirements. While traditional topology designs often emphasize scalability, they are typically limited by the necessity for dedicated switches to manage server connections. The development of software technologies that distinguish server and switch roles offers a unique opportunity to reconsider design priorities, paving the way for a more balanced assessment of scalability, energy efficiency, and infrastructure costs. Moreover, certain network topologies fail to be cost-effective due to their structural intricacies, often requiring far more node connections than those that are practically necessary. To address these challenges, we introduce VacoNet: a new flexible data center network topology that organizes nodes into structurally similar clusters, interconnected by a novel physical structure algorithm. Boasting high bisection bandwidth, VacoNet delivers robust network capacity, even when encountering bottlenecks. Furthermore, to connect a given set of nodes, VacoNet uses a minimal number of cables and switches, thereby drastically reducing both infrastructure costs and energy consumption. Simulation results show that VacoNet can reduce error rates by 20% and slash infrastructure costs by 70% compared to existing solutions. Additionally, it performs tasks 30% faster, underscoring its superior performance.

INDEX TERMS Data center, network topology, flexible connection, energy consumption, infrastructure cost.

I. INTRODUCTION

Data centers have become pivotal for meeting enterprise computing demands, offering robust infrastructure for cloud computing services. They also facilitate intricate communications between a vast array of computing resources [1]. A recent industry report highlights that over the past five years, the U.S. data center market has seen an expenditure surge of $ 23 billion USD [2]. Key design parameters such as scalability and data availability are critical for shaping the data center network topology, and they play a key role in minimizing deployment and maintenance costs [3].

Infrastructure cost is particularly influential, as it directly affects the initial capital investment and, consequently, the profitability of the data center [4]. Additionally, energy consumption is a substantial determinant of data center performance [5]. Therefore, an efficient data center topology design must consider a multitude of factors, including high energy-efficiency, low infrastructure cost, enhanced scalability, reduced latency, and shorter Average Path Length (APL) [6]. Numerous topologies such as FatTree [7], Flecube [8], BCube [9], Novacube [10], and DCell [11] have been proposed in the literature. However, these models

The associate editor coordinating the review of this manuscript and approving it for publication was Yougan Chen.

are generally inflexible. Specifically, they cannot adapt to accommodate an arbitrary number of nodes (or servers)[1] due to inherent structural limitations [12].

As a consequence of these limitations, both developers and manufacturers often resort to utilizing more nodes than actually needed [13], [14]. The consequence of this excess is a rise in both costs and energy usage [15], primarily attributable to servers that remain unused. In [16], the authors delve into the unique complexities of Facebook's data center network, which accommodates multiple services, each with its own set of traffic patterns. Some of these patterns are less commonly explored in existing literature. This research has its limitations, notably in its packet capture and timestamping methodology, which narrows the scope of the study. In [17], the authors introduce a methodology aimed at reducing energy consumption in data centers. These facilities traditionally require a large number of network devices to meet the rising demand for cloud services. The proposed approach manages the activation of communication links and ports, using an algorithm that ensures network connectivity while optimizing for energy efficiency. Nonetheless, the methodology has its shortcomings, including its inability to decrease infrastructure costs and potential ineffectiveness in aligning energy consumption with network load.

On the other hand, the type of link connection used has a significant impact on both the node degree and the scale of switches required [18]. Therefore, before assessing a particular network topology, the article first explores the advantages and disadvantages of various types of links. In general, networks feature three types of connections as shown in Figure 1: the Primary Data Link (also referred to as the α link), the Secondary Control Link (or β link), and the Tertiary Backup Link (known as the γ link). A BCube network uses only β connection, while both α and β are used in DCell. As for the γ link, it is used in Fat-Tree or some multistage circuit-switching networks also known as Clos topology [19]. A γ link is generally used to connect intermediate switches without a direct connection between nodes. γ links are used in many topologies namely the Clos topology [20] and Fat-Tree. However, they have low scalability that makes the γ link not suited for big data centers.While lambda topologies have been considered for data center networks, they may not be the most suitable option for high-performance requirements. Issues such as limited scalability and higher latency make lambda topologies less efficient for data center applications [6]. An α link is one of the most efficient connections between nodes with a maximally allowed bandwidth. The β link provides multiple non-blocking paths between the connected nodes. However, it needs an intermediate switch for communication. Thus, the use of β links combined with small-port-count switches is a good tradeoff between cost and performance. The design of BCube has originated from such an idea. The β link can be considered as the key building block of a massive data center network. The γ links have been used in FatTree and Clos topology to connect only switches, which reduce the number of connected nodes in the network [21]. The α link is a simple and direct connection that minimizes the intermediate buffering and improves the allowed bandwidth. The β link uses an additional intermediate switch compared to an α link, allowing multiple pairs of nodes to share their communication channels. Therefore, the use of β links combined with small-port-count switches is a good tradeoff between the cost and the performance. In addition, the use of a flat network topology reduces the number of switches in a data center network, since it connects a node to a single switch instead of many [22].

[1]In this document, the terms "node" and "server" will be used interchangeably.

A. MOTIVATION AND CONTRIBUTION

This work presents VacoNet, a dynamic and efficient network topology designed to connect a varying number of servers using small port-count switches and specialized β links. VacoNet employs a unique algorithm to minimize unused nodes, effectively reducing costs and increasing overall network efficiency. The algorithm dynamically adjusts the number of ports on each switch to eliminate redundancy. Additionally, this algorithm aims to maximize the number of server clusters directly connected with at least one switch (denoted as **directly connected clusters**) while reducing the number of clusters that need more than one intermediate switch to be connected (denoted as **not directly connected clusters**). Furthermore, VacoNet employs fault-free and fault-tolerant adaptive traffic routing algorithms, which contribute to its robustness and resilience. Unlike BCube, which primarily focuses on static configurations, VacoNet's adaptability allows it to better handle varying network conditions, making it more scalable and efficient.

The primary contributions of this work are:

1) A new data center network topology that employs only β links and two-degree nodes with small-port-count switches to connect the required number of servers.
2) An algorithm for determining the exact number of ports per switch to minimize unused nodes and wires, hence optimizing the network's cost and power consumption.
3) An algorithm to build the physical structure of the proposed topology. This algorithm defines the optimal connection pattern to reduce redundant connections between clusters, which optimizes the network in terms of APL, network latency, and throughput.
4) Two adaptive routing algorithms in both fault-free and fault-tolerant cases. Each node saves the connectivity status of its nearby nodes and uses it to derive a feasible routing paths then sends the data to their destination.

The remainder of this paper is organized as follows. Section II discusses related work. Sections III and IV define the proposed topology and identify its important characteristics. Sections V and VI, respectively, explain the routing strategies and experimental results. Finally, the conclusion is presented in Section VII.

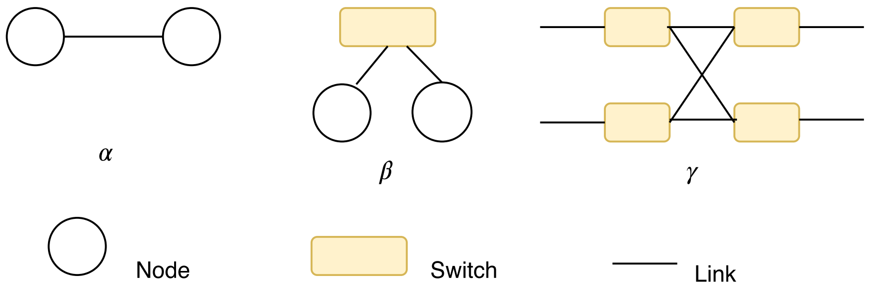

FIGURE 1. The α, β, γ connection.

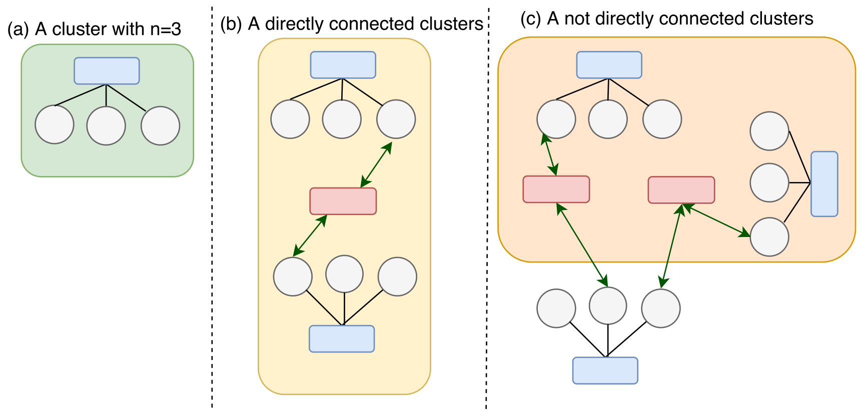
(a) A cluster with n=3
(b) A directly connected clusters
(c) A not directly connected clusters

FIGURE 2. An example of clusters connection.

II. RELATED WORK

Several topologies have been proposed in the literature for interconnecting nodes in data centers, including VL2, Clos topology, FatTree, BCube, and DCell. There are two types of topologies: tree-based topologies and recursive topologies.

A. TREE-BASED TOPOLOGIES

Tree-based topologies allow for scaling by adding more switch levels, and the bottom-level switches are responsible for connecting the servers to the network. The topology is designed in this category using typical tree-based topologies.

- VL2 is proposed to overcome critical issues in the data center network including oversubscription and fault-tolerance. By the use of virtual machines, VL2 enhances availability in the event of link or hardware failures. However, VL2 selects the intermediate switch at random, which is inefficient, especially when two nodes connected to the same edge switch need to communicate.
- Clos topology are a type of data center network topology that uses three levels of switches: Top of the Rack (ToR), intermediate, and aggregation switches. The number of ports per switch is the same for the intermediate and aggregation switches (n), and it is not limited to a ToR switch. This topology is able to connect $n_{ToR} \times \frac{n^2}{4}$ where n_{ToR} is the port count on each ToR switch.
- FatTree is an extension of the Tree topology. Unlike Clos topology, the same type of switches is used for all the levels. FatTree is a simple topology where high-performance switches are not required. However, FatTree has a scalability issue since the number of nodes is limited by the number of switch ports.
- Jellyfish uses ToR switch to connect the nodes based on random graph [23]. It exponentially increases the number of nodes while decreasing latency compared to a Fat-tree. Jellyfish uses a ToR switch with n ports, and r ports are used to connect it to other ToR switches. Jellyfish can interconnect $N(n-r)$ nodes for a network with N racks.

B. RECURSIVE TOPOLOGIES

Recursive topologies exploit lower-level structures to construct higher-level structures. Contrary to tree-based topologies, servers in recursive topologies can be connected to other servers or switches of different levels.This category includes:

- ScalNet is a network design for data centers that is highly scalable [24]. The ScalNet's first layer is composed of n servers and one n-port switch. The second layer is composed of $\frac{n^3}{2}$ of 1 layer ScalNet. This topology can connect a large number of nodes that reach $\frac{n^4}{2}$ nodes. However, ScalNet suffers from a high wiring complexity.

- DCell is a recursive topology in which the servers are connected via multiple ports. Each server is connected to multiple other servers via communication links via a single mini-switch. DCell_0 is the basic element of DCell composed of n servers and one n-port switch. Each server in a DCell_0 is connected to a switch in the same DCell_0 using a network cable.
  The initial stage is to assemble a DCell_1 from a collection of DCell_0s. Each DCell_1 has $n+1$ DCell_0s, and each DCell_0's server is linked to a server in another DCell_0, in that order. As a result, the DCell_0s are linked together, with one link connected each pair of DCell_0s. To make a DCell_k from numerous DCell_{k-1}s, a similar method is applied. Each server in a DCell_k will eventually have $k+1$ links used in each node where the first link is connected to a switch in DCell_0, and level_i link is connected to a node in the same DCell_i.
- BCube is a server-Centric topology where BCube_1 is composed of n BCube_0 and n n-port switches. Recursively, BCube_k is composed of n BCube_{k-1} and n^k extra n-port switches connected to exactly one node in each BCube_{k-1}. BCube requires more switches when constructing higher-level structures compared with DCell which uses only level_0 n-port switches. However, both BCube and DCell require servers to have $(k+1)$ network interface cards (NICs) to be involved in switching packets in the network. In a BCube structure, switches do not even connect to each other directly. Instead, they just forward. Many wires and switches are used in the BCube. It has a lot of complicated wiring that prevents it from being used beyond a shipping container based modular data center (MDC).

C. SUMMARY

Table 1 presents a comparison between FatTree, VL2, DCell and BCube. First, it can be seen that the scalability of VL2 and FatTree relies entirely on n and for a three-layer network, they can only connect $\frac{n^3}{4}$ nodes. Table 2 shows the number of unused nodes for some topologies. The price of an Ethernet switch port is fixed to 450 \$, and each inter rack cable costs 50\$, and a switch port consumes 12 watts of energy [25]. This Table reveals that the scalability of the selected topologies depends on the number of ports per switch.

The number of linked nodes can be defined in terms of each node's degree and the number of ports on each switch. Thus, in order to increase the scalability of massive data centers, both the port count per switch and the layer number must be increased. The purpose of this paper is to propose a novel data center interconnection network design, termed VacoNet, that scales dynamically utilizing commodity switches. VacoNet's performance has been enhanced with the addition of a new physical structure and routing algorithms. These steps are also illustrated in figure 3.

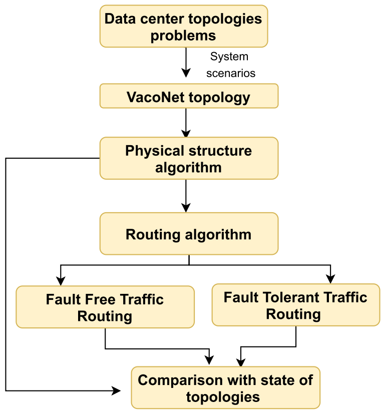
Data center topologies problems
System scenarios
VacoNet topology
Physical structure algorithm
Routing algorithm
Fault Free Traffic Routing
Fault Tolerant Traffic Routing
Comparison with state of topologies

FIGURE 3. The road map of the paper.

III. VACONET TOPOLOGY

A. MOTIVATION

A data center network's connection topology has a significant impact on its performance in terms of latency and average path length. Some proposed topologies such as FlatNet have ineffective connection patterns such as redundant cluster connections [26]. DCell has a double exponential scalability with a high variety of wiring. A pair of long-distance nodes can communicate to each other directly because each layer of the network is connected to the next. This topology minimizes the diameter of the entire network, but complicates wiring. As a result, building a DCell data center becomes highly complex. The major downside of BCube is that when using switches with a low port count, the number of nodes rises only by a factor of n for an n-port switch.

The previous facts prompted us to develop a novel cost-effective and scalable network topology termed VacoNet, which scales from 1 to $\frac{n^4}{2}$ nodes, where n is the number of ports per switch. VacoNet provides a solution of server consolidation while simultaneously reducing infrastructure costs and energy consumption.

B. PHYSICAL STRUCTURE

1) VACONET CONNECTIVITY

VacoNet topology is composed of two layers, the first layer (layer-1) is composed of n_1 nodes interconnected using one n_1-port switches. The second layer (layer-2) numbered from 1 to n_2 (n_1 and n_2 are computed in section III-B.2) Algorithm 1 is presented to define the network connection pattern for VacoNet in order to optimize direct connections between clusters (See figure 2 (b)) and minimize not directly connected clusters (figure 2 (c)) while preserving network scalability. i.e., Algorithm 1 ensures that a high number of servers are connected without using a large number of ports on switches and while minimizing the number of unused nodes. It's important to note that in VacoNet's topology, trunk ports used for interconnecting switches are distinct from the ports available for nodes. These trunk ports are specifically designed to facilitate high-speed data transfer

TABLE 1. Cost comparison between different topologies.

	BCube			FlatNet			FatTree			ScalNet		
Nodes	W$_n$	W$_{eg}$	W$_c$	W$_n$	W$_{eg}$	W$_c$	W$_n$	W$_{eg}$	W$_c$	W$_n$	W$_{eg}$	W$_c$
45	4	84	3550	19	192	7300	9	72	4950	83	1992	149×10^3
450	34	528	23200	62	192	13400	100	156	10850	198	4752	356×10^3
4500	124	1632	73600	413	4896	679200	420	5184	530400	500	12×10^3	450×10^3
45000	369	5112	228600	1659	19872	4.6×10^6	1299	15732	2×10^6	7488	179×10^3	6×10^6

TABLE 2. Comparison of some data center network topologies.

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

between switches and are not included in the count of available ports for nodes.

The servers in VacoNet are first grouped into clusters and interconnected with n-port switches. These clusters are linked to one another via switches referred to as internal switches.[2] This terminology is consistent with the schematics presented in figure 4, where external switches facilitate inter-layer connections and internal switches manage intra-layer connections.

The proposed interconnection is represented as a $(n_1 \times n_2)$ matrix denoted by L such that $L(i, j)$ ($\forall i \in \{1, .., n_1\}$ and $\forall j \in \{1, .., n_2\}$) is the index of the internal switch to witch node (i, j) (i.e., node j in cluster i) is connected to. Each row i of matrix L corresponds to one cluster and each column j presents the set of connections of each cluster through the node index j in the same cluster. In particular, the first column shows the connection of all the clusters using their first node (See figure 4). The numbers in the matrix $L(i, j)$ refer to the index of the used internal switch. The first row $L1$ of L is generated based on algorithm 1, while the subsequent rows are deduced sequentially from the row immediately preceding them by adding 1 modulo the number of rows as follows:

$$\forall i \in \{1..n_1\}$$
$$\forall j \in \{1..n_2\}$$
$$L(i, j) = (L(i-1, j) + 1) \; mod \; n_2^2 \tag{1}$$

Algorithm 1 initializes the first column of the first row of L_1 of the internal switch number to 1. The remaining columns are iteratively defined with the aim of expanding the number of clusters associated with the initial cluster. For each remaining column j of $L1$, the algorithm determines which intermediate switch resulted in the greatest number of connected clusters to cluster number 1.

A maximum of $i(i-1)$ clusters can be directly connected to each other by specifying the j^{th} column of L_1 and assuming that each switch has only i ports. Thus, in order to determine

[2]An 'internal switch' is used to connect nodes in different 1-layer VacoNet, while an 'external switch' is used to connect nodes within the same switch.

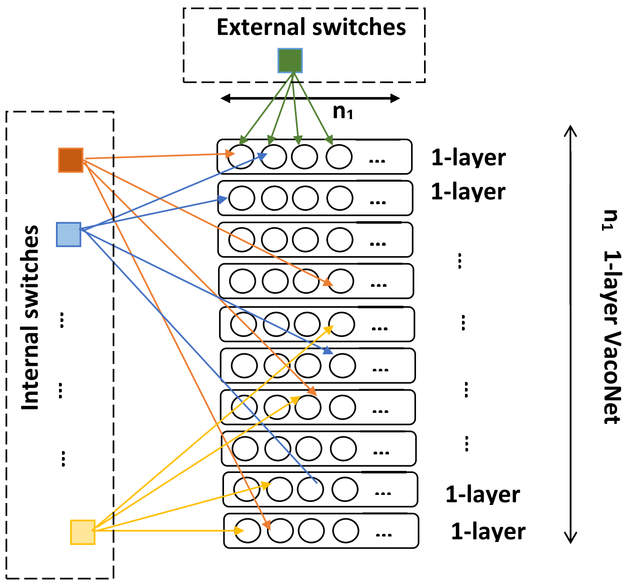

FIGURE 4. The connection pattern of the first and second layers.

whether or not the selected switch is optimal, the size of the connected clusters set LC is constantly compared to the value of $i(i-1)$ to verify that the selected switch is optimal. To formally model this, consider the matrix L defined as:

$$\forall i \in \{2..n_1\}, \forall j \in \{1..n_2\}$$
$$L(i, j) = (L(i-1, j) + 1) \quad \text{mod} \quad n_2. \tag{2}$$

The matrix L captures the complex network of connections between the clusters. The equation 2 provides a systematic way to generate these connections, and the resulting matrix L is used to assess whether the connected clusters set LC reaches the theoretical maximum of $i(i-1)$, thereby verifying the optimality of the selected switch.

The function *Connected Clusters* is proposed to find the best internal switch to be used from $\forall j \in \{1, .., n_2\}$. So, at each step i ($\forall i \in \{2, .., n_1\}$), the algorithm checks the number of connected clusters if it is maximized with

i-port switches (i.e., $i(i-1)$), then it is directly selected. After generating L_1, the function *LinkedClusters* computes the set LC that presents linked clusters distances so that $\forall i \in \{1..n_1\}$, $\forall j \in LC$, the clusters i and $(i+j)$ *mod* n_2 are directly connected. Figure 5 shows an example of Vaconet connection using 3-port switches. This figure highlights the arrangement of nodes and the role of internal switches in connecting them. The number of connected nodes is equal to 36, grouped into 12 clusters, interconnected using 12 switch internal.

Algorithm 1 Linked Clusters Maximization Algorithm

```
0: procedure LCC($n_1$,$n_2$)
0: $Cv$ is the connectivity vector.
0: $j_{selected}$ is the selected switch number to be insert in $L_1$.
0: Input:
0: $n_1$ is the number of column of $L$.
0: $n_2$ is the number of row number of $L$.
0: Output:
0: $L_1$ is the first row of $L$.
0: $L_1[1] \leftarrow 1, j_{selected} \leftarrow 0$;
0: for $i \leftarrow 2$ to $n_1$ do
0:     $Cv[\ ] \leftarrow \emptyset$
0:     for $j \leftarrow 1$ to $n_2$ do
0:         $L_1(i) \leftarrow L_1(i-1)+j$
0:         $Cv(j) \leftarrow ConnectedClusters$(i,$L_1$)
0:         if $Cv(j) \leftarrow i(i-1)$ then then
0:             Break
0:         end if
0:     end for
0:     $j_{selected} \leftarrow \text{argmax}(Cv)$
0:     $L_1(i) \leftarrow L_1(i-1)+j_{selected}$
0: end for
0: function ConnectedClusters(p,$L_1$)
0:     Input:
0:     $L_1$ is the first row of $L$
0:     $p$ is the index of the switch internal
0:     Output:
0:     $LC$ is the vector of the connected cluster
0:     $LC[\ ] \leftarrow 0$
0:     $k \leftarrow 1$
0:     for $i \leftarrow$ p down to 1 do
0:         for $j \leftarrow 1$ to $i$-1 do
0:             $LC(k) \leftarrow L_1(i) - L_1(i-j)$
0:             $k \leftarrow k$+1;
0:         end for
0:     end for
0:     $LC \leftarrow$unique (mod([LC   $n_1^2$-LC], $n_2$))
0:     return (Length($LC$))
0: end function
```

To analyze the complexity of the Algorithm 1, the condition in L.13 and the *break* in L. 16 should be investigated when they occur, which is very challenging to compute. Therefore, an upper bound on the complexity is computed by disregarding the break condition in L. 16. By neglecting the complexity of the instruction L. 13 computed to instruction

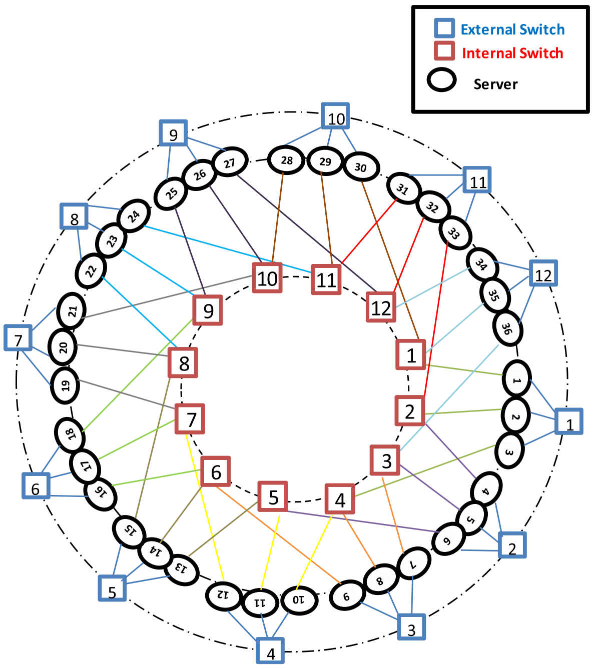

FIGURE 5. An example of VacoNet connection for 3-port switches.

in L. 14, the complexity of *Linked Clusters Maximization algorithm (LCM)* can be approximated by:

$$C_{LCM} \approx \sum_{i=2}^{n_1}\sum_{j=1}^{n_2} C_{cc}(i) + \mathcal{O}\left(n_2 \log_2(n_2)\right) \quad (3)$$

where $O\left(n_2 \log_2(n_2)\right)$ is the complexity of *argmax* operation in L. 19 assuming that the condition in L. 15 was never satisfied, i.e. the vector C_v contains n_2 elements to sort. C_{cc} is the complexity of the function *Connected cluster* which is equal to:

$$\begin{aligned} C_{cc} &= \sum_{i=2}^{n_1}\sum_{j=1}^{n_2} \mathcal{O}\left(\frac{i^2}{2}\right) \\ &= \sum_{j=1}^{n_2} \frac{1}{2}\mathcal{O}\left(\sum_{i=2}^{n_1} i^2\right) \\ &\approx n_2\mathcal{O}\left(\frac{n_1^2}{2}\right) \end{aligned} \quad (4)$$

Consequently, C_{LCM} can be written as:

$$\begin{aligned} C_{LCM} &\leq \sum_{j=2}^{n_2}\sum_{i=2}^{n_1} \mathcal{O}\left(\frac{i^2}{2} + \mathcal{O}\left(n_2 \log_2(n_2)\right)\right) \\ &\approx n_2\mathcal{O}\left(\frac{n_1^2}{2}\right) + \mathcal{O}(n_2 \log_2(n_2)) \\ &\approx \mathcal{O}\left(n_2\frac{n_1^2}{2} + \log_2(n_2)\right). \end{aligned} \quad (5)$$

Thus the complexity of the proposed topology highly depends on n_1 (number of nodes in layer-1) and n_2 (number of ports per switch), which are typically lower than the state of art techniques since VacoNet aim is to reduce the unsued number of nodes. This proves that the complexity of the physical structure algorithm is still acceptable.

2) CONTROLLED VACONET

For a specific number of nodes, the algorithm 2 adjustes the size of L according to the operator's needs. Given the needed number of nodes n_{serv}, the n-port switch n_2 is initialized to $Floor(\sqrt[3]{n_{serv}})$ and n_1 to $Ceil(\frac{n_{serv}}{n_1})$, then, based on their previous values n_1 and n_2 are iteratively updated. So, if the number of rows n_1 in the matrix L is bigger than $\frac{n_2^3}{2}$, then n_2 will be increased by 1. So, the number of rows in L (i.e. $n_1 = n_2^2$) will be increased. Otherwise, the number of n-port switch (n_2) is the same while n_1 is increased by 1. Figure 6 shows an example for VacoNet topology for 8 nodes.

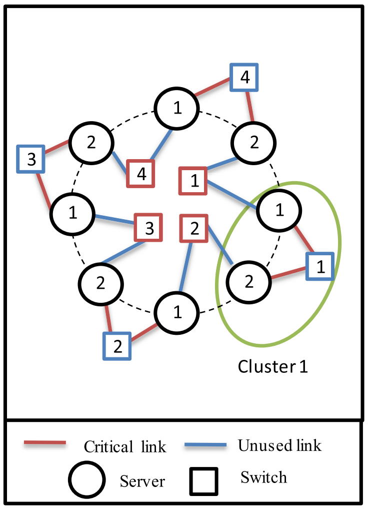

FIGURE 6. VacoNet topology for 3 port switch.

Algorithm 2 n-Port switch(n_{serv})
```
0: $n_1$: the number of columns in $L$.
0: $n_2$: the number of rows in $L$.
0: $n_2 \leftarrow Floor(\sqrt[3]{n_{serv}})$
0: $n_1 \leftarrow Ceil(\frac{n_{serv}}{n_1})$
0: if $Or(n_1 > \frac{n_2^3}{2}, n_1 < n_2^2)$ then
0:     $n_2 \leftarrow n_2 + 1$
0:     $n_1 \leftarrow n_2^2$
0: end if
0: return $n_1, n_2$
```

IV. KEY FEATURES

In this section, some key features of the VacoNet topology will explored and analyzed in comparison with other topologies in terms of network latency, cost reduction, power consumption and average path length.

A. NETWORK LATENCY

Network latency is an important parameter too. It is the sum of the transmission delay, the queuing delay for each hop, and the propagation delay. Hence, the latency of a node i sending a message to node j can be written as:

$$\begin{aligned} Latency_{i,j} = {} & d_{qb}^{sr} N_{sr}^{i,j} + d_{qb}^{sw} N_{sw}^{i,j} + d_T N_{Lk}^{i,j} \\ & + N_{sw} d_p^{sw} + N_{sr} d_p^{sr} + d_P. \end{aligned} \tag{6}$$

where d_{qb}^{sw} and d_{qb}^{sr} are the queuing delay of switches and servers. d_p^{sw} and d_p^{sr} are the propagation delay of switches and servers, respectively. d_P and d_T are the propagation and the transmission delays for 1 link, respectively. $N_{Lk}^{i,j}$ is the number of links between nodes i and j. $N_{sw}^{i,j}$ and $N_{sr}^{i,j}$ are the total number of switches and servers respectively.The number $N_{sw}^{i,j}$ is equal to:

$$N_{sw}^{i,j} = N_{sr}^{i,j} = PL_{i,j}, \tag{7}$$

$$N_{links} = 2PL_{i,j}. \tag{8}$$

where $PL_{i,j}$ is the path length between the sender node i and the receiver node j. Thus, Eq.6 becomes:

$$\begin{aligned} Latency_{i,j} = {} & d_{qb}^{sr}.PL_{i,j} + d_{qb}^{sw}.PL_{i,j} + d_T.2PL_{i,j} + N_{sw} d_p^{sw} \\ & + N_{sr} d_p^{sr} + d_P \\ = {} & PL_{i,j}(d_{qb}^{sr} + d_{qb}^{sw} + 2d_T) + N_{sw} d_p^{sw} + N_{sr} d_p^{sr} \\ & + d_P. \end{aligned} \tag{9}$$

Based on 9, the average latency can be expressed as:

$$\begin{aligned} Latency = {} & APL(d_{qb}^{sr} + d_{qb}^{sw} + 2d_T) + d_P \\ & + N_{sw} d_p^{sw} + N_{sr} d_p^{sr}. \end{aligned} \tag{10}$$

According to Eq.10, the average latency is proportional to the APL. So, the smaller the APL, the smaller is the latency. Given that VacoNet reduces the APL compared with all previous topologies, the latency will be reduced. Therefore, the proposed topology improves the network in terms of latency, which is a fundamental feature that enables data centers to provide faster services.

B. COST REDUCTION

The infrastructure cost is a critical parameter in designing data center topologies. The $Cost_T$ is the total cost of a topology can be expressed as:

$$Cost_T = Cost(Cables) + Cost(switches). \tag{11}$$

The cabling cost $Cost(Cables)$ can be computed according to the total used number of cables N_{cb} and the cost per cable C_{cb} such that $Cost(Cables) = N_{cb} \times C_{cb}$ and $N_{cb} = n \times N_{sw}$. The cost of switches is equal to $Cost(switches) = n \times N_{sw}$ where N_{sw} denotes the total number of switches. VacoNet connects the needed number of servers which reduces considerably network's cost.

C. POWER CONSUMPTION

The proposed topology is able to reduce the number of unused nodes in the network which increases the energy saving. So, let $p_{i,j}^{sv}$ and $p_{i,j}^{sw}$ be the consumed energy by the i^{th} port of the j^{th} server and the j^{th} switch, respectively. Let k be the number of ports per server and n the number of ports per switch. Let p_o^{sw} be the switch power overhead (including switch fans, line cards) and p_o^{sv} the server power overhead. Thus, the total consumed energy E is equal to:

$$E = \sum_{i=1}^{N_{sw}} \sum_{j=1}^{n} p_{i,j}^{sw} + N_{sw} p_o^{sw} + \sum_{i=1}^{N_{sv}} \sum_{j=1}^{k} p_{i,j}^{sv} + N_{sv} p_o^{sv}, \qquad (12)$$

The power overhead per server and switch is assumed to be the same and denoted by P_o. For an active network where all the ports consume the same energy per port P, the network energy consumption is:

$$E^{FA} = (N_{sw}n + N_{sv}k)P + (N_{sw} + N_{sv})P_o. \qquad (13)$$

Therefore, for more energy saving, the objective is to find a set of optimal N_{sw} and N_{sv} that reduces as much as possible the number of unused number of nodes which minimizes P_a.

D. SCALABILITY AND COST PER SERVER

VacoNet connects up to $O(n^4)$ nodes, which is bigger than $O(n^2)$ for DCell and BCube, and bigger than $O(n^3)$ for FlatNet while maintaining the same number of cables and switches per node as FlatNet and BCube.

E. THE AVERAGE THROUGHPUT

The throughput of a network is determined by packet delay, the data rate of the channel, and the rate of successfully received messages. Due to the significant decrease in latency in VacoNet, the average throughput is also decreased in comparison to DCell, FatTree, BCube, and FlatNet, which means that VacoNet is able to deliver a higher number of messages compared with all the other topologies. This is justified by the fact that VacoNet uses only the needed number of switches and servers, allowing to reduce the number of intermediate hops when sending a packet to its destination.

F. PROTOCOL COMPATIBILITY AND REAL-WORLD APPLICABILITY

In terms of network setup, VacoNet's adaptability to existing norms and protocols is a key advantage. It is easily compatible with both Virtual Local Area Networks (VLANs) and Private VLANs (PVLANs), enhancing security and network compartmentalization. Additionally, VacoNet accommodates encapsulation protocols like IEEE 802.1Q, facilitating packet prioritization and tagging. Furthermore, it's essential to consider the various versions of the Spanning Tree Protocol (STP), including PVSTP+, RSTP, RPVST+, and MSTP. The choice of STP version can significantly impact network performance. Given these versions and their specific enhancements, VacoNet's adaptability is underscored, aligning it with real-world networking scenarios and making it not only innovative but also highly versatile in practical conditions.

Furthermore, in evaluating VacoNet's architecture, it's pertinent to align it with established industry standards, such as the TIER classification system. These TIER levels, ranging from 1 to 4, serve as benchmarks for component redundancy and overall operational reliability in data centers. VacoNet's design, characterized by its robust fault tolerance and adaptive routing mechanisms, naturally aligns with the rigorous criteria set forth for TIER 3 data centers. This classification not only underscores VacoNet's high reliability but also contextualizes the performance metrics discussed in this paper.

V. ROUTING SCHEME

In the following section, we delve into the intricacies of our proposed Routing Scheme, a crucial component for the efficient and reliable transmission of data within the network. This scheme is specifically designed to optimize path selection, reduce latency, and ensure scalability while maintaining robust security features. We have divided this section into two distinct subsections for a comprehensive understanding: 'Fault-Free Routing' and 'Fault Tolerance'.

A. FAULT FREE TRAFFIC ROUTING

VacoNet forwards the packets from the source (S_2, S_1) to the destination (D_2, D_1) according to the values of Ω. The proposed Algorithm 5 returns a path of maximum 10 hops. Firstly, algorithm 3 generates the matrix Ω_k which represents the vector of clusters connected to the first cluster via exactly $(k + 1)$ internal switches. Function *setdiff* makes sure that Ω_k does not contains any repeated value. The index vector of all the items in Ω_k is saved in Cxt matrix. For the VacoNet network shown in Fig 6, the vector Ω contains two elements $\Omega_0 = \{1, 3\}$ and $\Omega_1 = \{2\}$, meaning that cluster 1 and clusters $\Omega_0 + 1$ (2 and 4) are directly connected, but cluster 1 and Ω_1 are not (cluster 3 and 1 are connected via 2 internal switches).

Subsequently, for each entry in the connected cluster vector (Ω), Algorithm 4 looks for the connecting internal switch, and saves its index (c_1, c_2) in the matrix of intermediate internal switches denoted by Cx. For the example presented in Fig 6, Algorithm 4 output is showed in Figure 7.

Let us consider the case of routing a packet from source (1,2) to destination (4,1) in a 64-server VacoNet. $(1, 2) \rightarrow (4, 1)$ matches the case $(mod(4-1, 4) \in \Omega_0)$ which represents the scenario where there is a direct connection between the source and destination subsystems and the route is $(1, 2) \rightarrow (1, 1) \rightarrow (4, 2) \rightarrow (4, 1)$.

The computation complexity of Algorithm. 5 depends on whether $(S_2 = D_2)$ or not. When $(S_2 = D_2)$, the algorithm complexity can be approximated by $\mathcal{O}(1)$ since only one instruction will be executed (L. 14). When $(S_2 \neq D_2)$, then the routing algorithm has to compute the required path to forward the packet from the source to the destination. In this case, the instructions from (L.16 to L. 35) should be executed. Note that all of these instructions represent only variables assignments to create the adequate variables T and L that will be required later for the path creation. So, they do not have high computation resources. Therefore, when ($S_2 \neq D_2$), the computation complexity can be approximated by the complexity of the instructions from (L29 to L34). The number of times the instructions from (L29 to L34) will be repeated depends on k and the length of Cxt_k, so it is very hard to get an accurate estimation of the number of occurrences of these instructions. However, it can be seen that each repetition of instructions (L29 to L34), creates an intermediate node in the path. So, these instructions will be repeated around PL times. So, their complexity can be approximated by $\mathcal{O}(PL)$. Consequently, the complexity of the routing algorithm can be written as:

$$C_{Rt} = \mathcal{O}(1)\, P(S_2 = D_2) + \mathcal{O}(PL)\, P(S_2 \neq D_2) \leq \mathcal{O}(PL) \tag{14}$$

where $P(S_2 = D_2)$ is the probability of ($S_2 = D_2$) and $P(S_2 \neq D_2)$ is the probability of ($S_2 \neq D_2$).

$$C_{Rt} \leq \mathcal{O}(PL)\,. \tag{15}$$

The complexity of the routing algorithm depends on the path which is linearly proportional to the actual path length from the source to the destination. On the other hand, VacoNet reduces the path length in the network compared to the other topologies, this means that the complexity of the routing algorithm is acceptable and does not require very high computational resources.

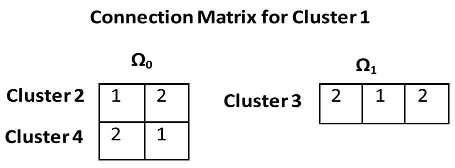

FIGURE 7. ***Generation of the matrix Cx* output for cluster 1.**

B. FAULT TOLERANT TRAFFIC ROUTING

Algorithm 6 is proposed to be used in the fault tolerant case to find alternative paths through the various clusters of the network. In the algorithm, *MaxLifeTime* represent the maximum lifespan of a broadcast packet in VacoNet. The fault tolerant traffic routing algorithm allows the system to use a reachable server as new source in case of a fault link transmission and forwards packets with a maximum number of hops *MaxLifeTime*. The *routing data packets* procedure is modified such that in the case of link failure it looks for all possible paths using available switches in *idx*, and uses them to resume the routing. The modified *for* loop α of Algorithm 6 is shown in the code fragment Algorithm 7. This algorithm is designed to optimize network routing by dynamically adjusting paths when a link failure is detected. Algorithm 7 plays a crucial role in maintaining high availability and robustness. The algorithm takes as input the current network topology and the status of each link, and it outputs an optimized routing table that avoids failed links. In terms of computational complexity, the fault tolerant algorithm operates operates with a time complexity of $O(n \log n)$ in the average case, where n is the number of

Algorithm 3 Generation of the Matrix Ω_k (Ω)

```
0: $\Omega_U$ is Union of all the $\Omega_k$ vectors
0: $Cxt$ is the intermediate switch connection
0: $\Omega_k$:the vector of clusters connected to the first cluster via
   exactly k+1 internal switches
0: Input:
0: $\Omega$ is the vector of directly connected clusters
0: Output:
0: $Cxt$ is the intermediate switch connection
0: k$\leftarrow$ 0
0: $\Omega_0 \leftarrow \Omega$
0: $\Omega_U \leftarrow \Omega_0$
0: $Cxt_0 \leftarrow$ 1 to length($\Omega_0$)
0: **while** length($\Omega_U$) $< n_2$ **do**
0:     k$\leftarrow$k+1
0:     id$\leftarrow$0
0:     **for** i$\leftarrow$ 1 to length($\Omega_{k-1}$) **do**
0:         **for** j$\leftarrow$ 1 to length($\Omega_0$) **do**
0:             **for** s$\leftarrow$ 1 to length($Cxt_k$) **do**
0:                 id$\leftarrow$ id+1
0:                 $\Omega_k(id) \leftarrow mod(\Omega_{k-1}(i) + \Omega_0(j), n_2)$
0:                 $\Omega_k(\Omega_k = 0) \leftarrow n_2$
0:                 $Cxt_k(id, s) \leftarrow j$
0:             **end for**
0:         **end for**
0:     **end for**
0:     $[\Omega_k, IA] \leftarrow setdiff(\Omega_k, \Omega_U)$
0:     **for** s$\leftarrow$ 1 to length($Cxt_k$) **do**
0:         $Cxt_k(s) \leftarrow Cxt_k(IA(s), s)$;
0:     **end for**
0:     $\Omega_U = unique([\Omega_U \Omega_k])$
0: **end while**
0: =0
```

Algorithm 4 Generation of the Matrix $Cx(\Omega)$

```
0: Input:
0: $\Omega$ is the vector of directly connected clusters
0: Output:
0: $Cx$ is the intermediate internal switch index
0: **for** $c_1 \leftarrow$ 1 to length $\Omega$ **do**
0:     **for** $c_2 \leftarrow$ 1 to $n_1$ **do**
0:         **if** $L(1, c_1) \leftarrow$ L($\Omega(i) + 1, c_2$) **then**
0:             Cx(i,1)$\leftarrow c_1$
0:             Cx(i,2)$\leftarrow c_2$
0:         **end if**
0:     **end for**
0: **end for**
0: =0
```

Algorithm 5 Routing Data Packets $((S_2, S_1), (D_2, D_1), Cxt, Cx, \Omega)$

```
0: $T_{i,1}$ is the first intermediate switch
0: $T_{i,2}$ is the second intermediate switch
0: $L_j$ is the line number in the matrix L
0: Input:
0: $Cx$ is the matrix of intermediate internal switch index
0: $Cxt$ is the matrix of intermediate switch connection
0: $\Omega$ is the vector of directly connected clusters
0: $(S_2, S_1)$ is the source coordinates
0: $(D_2, D_1)$ is the destination coordinates
0: $idx$ is the intermediate internal switch index
0: Output:
0: $Path$ is the path from the source to the destination
0: if $S_2 = D_2$ then
0:     $Path \leftarrow (S_2, S_1) \rightarrow (D_2, D_1)$
0: else
0:     $idx \leftarrow$(find($\Omega_k = mod(S_2 - D_2, n_2)$))
0:     for i←1 to k+1 do
0:         $T_{i,1} \leftarrow Cx(Cxt_k(idx, i),1)$
0:         $T_{i,2} \leftarrow Cx(Cxt_k(idx, i),2)$
0:     end for
0:     $L_0 \leftarrow S_2$
0:     for j←1 to k do
0:         $L_j \leftarrow (\Omega(Cxt_k(idx, j)) + L_{j-1}, n_2)$
0:     end for
0:     $L_{k+1} \leftarrow D_2$
0:     Path← $[S_2S_1]$
0:     for i←1 to k+1 do
0:         for s← 1 to length($Cxt_k$) do
0:             m←floor($\frac{i}{2}$)
0:             P1←floor($\frac{i+1}{2}$)
0:             P2←mod(i+1,2)+1
0:             Path←[Path,($L_m T_{P1,P2}$)]
0:         end for
0:     end for
0:     Path←Unique[Path, $(D_2, D_1)$]
0: end if
0: =0
```

Algorithm 6 Fault Tolerant Traffic Routing $(MaxLifeTime, \Omega)$

```
0: Input:
0: $MaxLifeTime$ is maximum number of hops
0: $\Omega$ is the vector of directly connected clusters
0: Output:
0: $Cxt$ is the intermediate switch connection
0: k ← 0
0: $\Omega_0 \leftarrow \Omega$
0: $\Omega_U \leftarrow \Omega_0$
0: $Cxt_0 \leftarrow$ 1 to length($\Omega_0$)
0: while k ≤ $MaxLifeTime$ do
0:     k← k+1
0:     id← 0
0:     for i← 1 to length($\Omega_{k-1}$) do
0:         for j← 1 to length($\Omega_0$) do
0:             for s← 1 to length($Cxt_k$) do
0:                 $id \leftarrow id + 1$
0:                 $\Omega_k(id) \leftarrow (\Omega_{k-1}(i) + \Omega_0(j), n_2)$
0:                 $\Omega_k(\Omega_k == 0) \leftarrow n_2$
0:                 $Cxt_k(id, s) \leftarrow j$
0:             end for
0:         end for
0:     end for
0: end while
0: =0
```

Algorithm 7 Dynamic Routing in Case of Link Failure

```
0: while $(T_{i,1}$ and $T_{i,2})\emptyset$ do
0:     idx← $(find(\Omega_k = mod(S_2 - D_2, n_2)))$
0:     for k← 1 to length (idx) do
0:         for i← 1 to k+1 do
0:             $T_{i,1} \leftarrow Cx(Cxt_k(idx, i),1)$
0:             $T_{i,2} \leftarrow Cx(Cxt_k(idx, i),2)$
0:         end for
0:     end for
0: end while
0: =0
```

nodes in the network. However, it's crucial to consider worst-case scenarios for a more comprehensive understanding of the algorithm's efficiency. In the worst-case, the algorithm exhibits a time complexity of $O(n^2)$, primarily due to the nested loops involved in the fault-tolerance mechanism. This worst-case complexity is still manageable for networks of moderate size but could become a bottleneck for extremely large-scale systems. Therefore, while the algorithm performs efficiently under typical conditions, future work could focus on optimizing its worst-case performance.

VI. SYSTEM EVALUATION

In this section, the simulation environment and the performance evaluation for Vaconet will be presented for different key features.

A. SIMULATION ENVIRONMENT

A DELL desktop computer with Windows 10 with 64-bit operating system is used to perform the simulations. The computer has a random access memory size of 32.00 gigabytes. In this simulation, the proposed and even the existent topologies may require switches with a specific number of ports to ensure the required network connectivity. Therefore, the cost needed to build these special switches may be higher than the standard ones. Consequently, fixed-configuration switches with the smallest number of ports that can accommodate the required number is used in the sequel. As detailed in [27], the adopted fixed-configuration cisco switches may accommodate 5, 8, 10, 16, 24, 28, 48, or 52 ports. Table 3 shows the configuration of the experiments.

TABLE 3. Simulation parameters.

Packet size	1024 bytes
The packet sending data rate	1 Mbps
The power overhead per switch/server	5 watt
The energy consumption per switch port	12 watt
The cost of an Ethernet switch port	450 \$
The cost of each inter rack cable	50 \$
Possible fixed number of ports per switch	5, 8, 10, 16, 24, 28, 48, 52

B. SIMULATION RESULTS

The performance of the proposed topology is investigated in terms of power consumption, infrastructure cost (cabling cost and switch cost), network capacity and average path length.

1) ENERGY CONSUMPTION

Fig 8 depicts the energy consumption of the proposed topology compared to FlatNet, BCube, FatTree, and ScalNet with a varied number of servers. It can be seen that VaConet consumes less power than the other topologies. Vaconet performs the data transmission only through the available nodes which decreases the consumed energy. However, the other topologies transmit the data through the unused nodes which explains the high values for the consumed energy. Even-though the investigated topologies in Fig 8 might require using switches with fixed port configurations higher than what is actually needed (as detailed in Table 3), the total power consumption remains almost unaffected since the added extra ports will be unused and will not consume any extra power. These extra ports can be useful however in connecting new servers whenever needed.

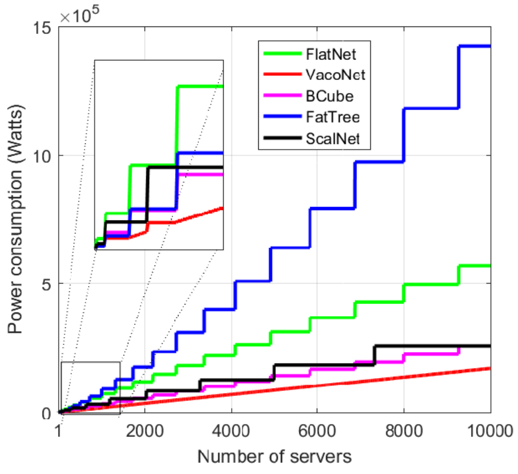

FIGURE 8. The power consumption of the proposed topology compared to FlatNet, BCube, FatTree and ScalNet.

2) INFRASTRUCTURE COST

Fig 9 highlights two crucial metrics for assessing the efficacy of data center topologies. Subfigure 9(a) compares the switch costs of VacoNet with those of FatTree, FlatNet, ScalNet, and BCube when the number of servers varies from 1 to 1,000.

TABLE 4. Cost comparison with topologies supporting 2 layers only.

	Vl2	Clos topology	Jellyfish	VacoNet
Number of nodes	$\frac{(n-2)n^2}{4}$	$\frac{n^2}{4} \times n_{ToR}$	$N(n-r)$	$n_1 \times n_2$
Number of switches	$\frac{3n}{2} + \frac{n^2}{4}$	$\frac{3n}{2} + \frac{n^2}{4}$	N	$2n_2$
Number of switch per node	$\frac{(n+6)}{(6n-2n)}$	$\frac{6+n}{n \times n_{ToR}}$	$\frac{1}{n-r}$	$\frac{2}{n_1}$
Network diameter	6	6	4	4

For a setup involving 8,200 servers, VacoNet emerges as the most cost-effective, boasting a cabling cost of 8.02×10^5, which is lower than FlatNet (9.26×10^5), BCube (1.38×10^6), and FatTree (2.24×10^6). Subfigure 9(b) delves into the switching costs associated with VacoNet, showing that it retains its cost-efficiency even when using switches with fixed-port configurations.

Note that any data center configuration will also require standard infrastructure elements like patch panels, racks, wall outlets, and jacks. Though not explicitly accounted for in our simulations, these elements are universally necessary in all data center setups. VacoNet significantly reduces the cabling complexity and the number of switches needed, thereby offering cost advantages. By focusing on these key aspects, VacoNet aims to provide a more efficient and cost-effective solution without compromising on performance or reliability.

3) AVERAGE PATH LENGTH

Fig 10 (a), (b) show VacoNet' APL vs the number of switches. Fig 10 (c) depicts the APL of the proposed topology compared with BCube, FlatNet, ScalNet and FatTree. The number of servers is varied between 1 to 1000. First, it can be seen that a small number of nodes both FlatNet and VacoNet outperform FatTree, ScalNet and BCube with shorter APL. However, for a bigger data center, VacoNet achieves a lower APL compared with FatTree FlatNet, BCube, and ScalNet. In fact, when the number of switches per server is increased, more alternatives paths will be generated, so, the packets transmission will be faster, which reduces the APL.

4) NETWORK CAPACITY

The network capacity is an important parameter for a data center. The higher the network capacity the better the network traffic. NC_{ABT} can be written as:

$$NC_{ABT} = \frac{1}{APL}. \tag{16}$$

Fig 11 shows network capacity of VaConet compared to other topologies with a varied number of servers. VacoNet is better than the other topologies with a network capacity equal to 0.22 for 1000 network nodes, which is less than 12 nodes BCube, 222 nodes FlatNet, and 1 node ScalNet.

To further validate the performance of the proposed topology, VacoNet is compared against Vl2, Clos topology, and Jellyfish. Table 4 shows a comparison of VacoNet with

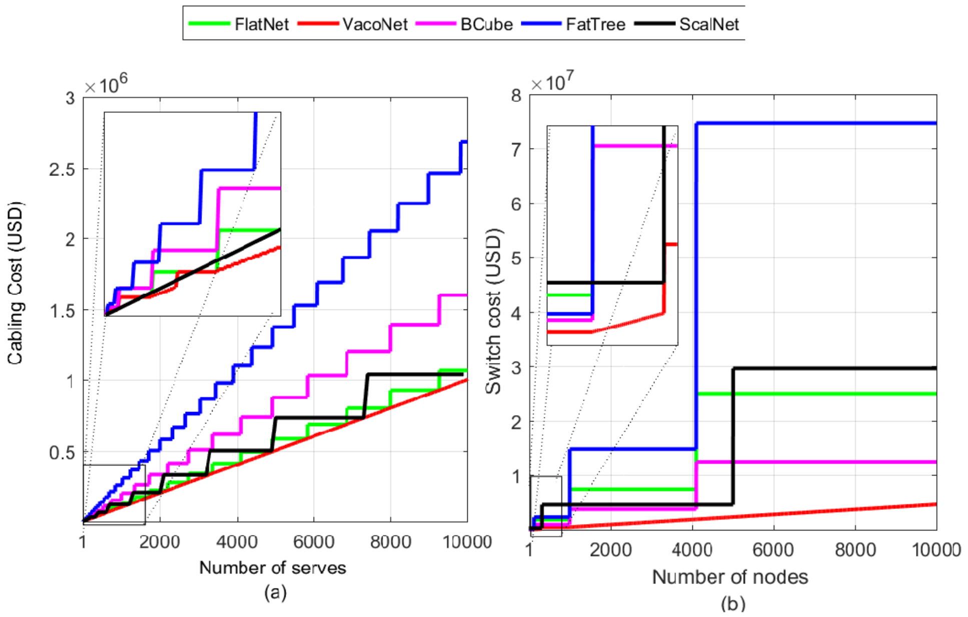

FIGURE 9. The switch and cabling cost of VacoNet compared with other topologies with a varied number of servers.

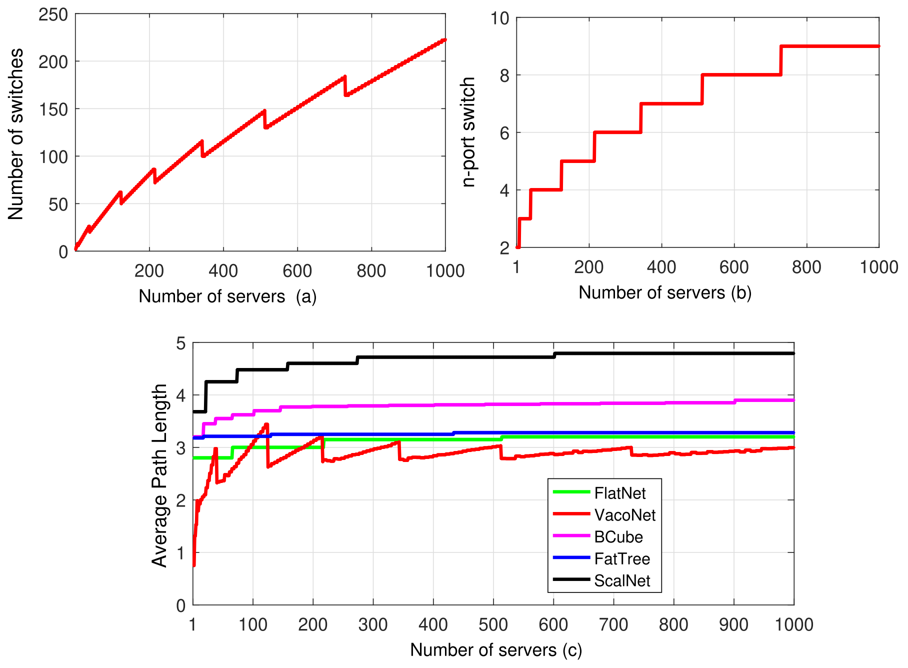

FIGURE 10. The average path length of VacoNet compared with FatTree FlatNet, BCube, and ScalNet.

other topologies where n is the number of ports per node. The performance of Clos topology and Jellyfish is highly influenced by their configuration parameters which are n_{ToR}, N_{sw}, and r. However, the number of nodes in VacoNet depends on the number of requested nodes in the topology to reduce the unused nodes.

5) THE CONNECTION FAILURE RATE

Fig 12 shows the connection failure rate in VacoNet as a function of link failure rate. The *MaxLifeTime* values are: 8, 10 and 12 hops, and varied link failure rate from 0.02 to 0.3. The number of servers is fixed to 1000.

The connection failure rate increases superlinearly with the link failure rate. Moreover, the new topology resists to the link failure even when the failure rate reaches 30%. This means that when for *MaxLifeTime* = 8hops (equal to the diameter) and even when one third of the links fail, the connection failure rate is only 15%. The connection failure rate is 9% when the link failure rate is 30% and the *MaxLifeTime* =12 hops. The proposed topology tries

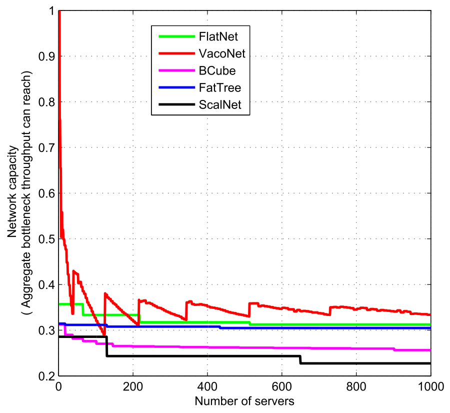

FIGURE 11. Network capacity comparison of VaConet, FlatNet, BCube, FatTree and ScalNet.

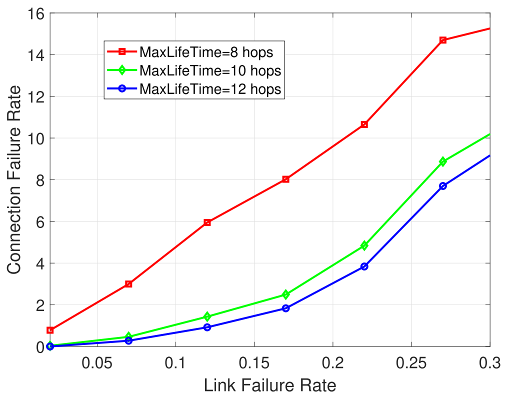

FIGURE 12. The connection failure rate of VacoNet.

to find the best path in terms of reducing the APL while respecting the *MaxLifeTime*. So, VacoNet can find the transmission path that optimizes the APL and offsets between different performance metrics. In addition, unlike traditional topologies, VacoNet is designed to have a higher number of alternative paths between nodes. This multiplicity of routes not only enhances the adaptability of the network but also significantly reduces the risk of connection failures. In scenarios where a particular path experiences a fault, VacoNet's adaptive routing algorithms can quickly reroute the data packets through an alternative path, thereby ensuring uninterrupted data flow. This abundance of alternative paths makes VacoNet more resistant to failures compared to other topologies, offering a distinct advantage in maintaining high levels of network performance and reliability.

VII. CONCLUSION

In this paper, a new data center network topology called VacoNet was presented. VacoNet reduced the number of unused nodes in the network and connected only the necessary ones. Thanks to its novel physical algorithm, the proposed topology computed the optimal number of ports per switch to minimize the number of unneeded nodes. VacoNet featured fault-free and fault-tolerant adaptive traffic routing algorithms for forwarding data to nodes. Using β links and small-port-count switches, VacoNet enhanced the scalability of a data center while maintaining service quality. VacoNet was characterized by its managed scalability, robust fault tolerance, and short average path length. Simulation results showed that VacoNet reduced both cost and energy consumption compared to state-of-the-art techniques while maintaining high network performance. In future work, we plan to account for the variable costs associated with differing cable lengths to provide a more accurate cost model for large-scale networks. There is also potential for optimizing VacoNet's energy efficiency and cost-effectiveness over extended periods of operation.

ACKNOWLEDGMENT

The statements made herein are solely the responsibility of the author[s].

REFERENCES

[1] W. Li, J. Liu, S. Wang, T. Zhang, S. Zou, J. Hu, W. Jiang, and J. Huang, "Survey on traffic management in data center network: From link layer to application layer," *IEEE Access*, vol. 9, pp. 38427–38456, 2021.

[2] 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.

[3] M. Shaukat, W. Alasmary, E. Alanazi, J. Shuja, S. A. Madani, and C.-H. Hsu, "Balanced energy-aware and fault-tolerant data center scheduling," *Sensors*, vol. 22, no. 4, p. 1482, Feb. 2022.

[4] A. Katal, S. Dahiya, and T. Choudhury, "Energy efficiency in cloud computing data centers: A survey on software technologies," *Cluster Comput.*, vol. 26, no. 3, pp. 1845–1875, Jun. 2023.

[5] C. L. Albarracín, S. Venkatesan, A. Y. Torres, P. Yánez-Moretta, and J. C. J. Vargas, "Exploration on cloud computing techniques and its energy concern," *Math. Statistician Eng. Appl.*, vol. 72, no. 1, pp. 749–758, 2023.

[6] Z. Chkirbene, R. Hadjidj, S. Foufou, and R. Hamila, "LaScaDa: A novel scalable topology for data center network," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2051–2064, Oct. 2020.

[7] B. Lan, F. Lei, K. Wu, and D. Dong, "DFR: Dynamic-thresold fault-tolerant routing for fat tree," in *Proc. 7th Asia–Pacific Workshop Netw.*, Jun. 2023, pp. 180–181.

[8] W. Xue, Z. Han, and X. Du, "Review of new data center network structure," in *Proc. 25th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2023, pp. 232–235.

[9] W. Fan, F. Xiao, H. Cai, X. Chen, and S. Yu, "Disjoint paths construction and fault-tolerant routing in BCube of data center networks," *IEEE Trans. Comput.*, vol. 72, no. 9, pp. 2467–2481, Sep. 2023.

[10] S. Rajeshwari and M. Rajesh, "On optimal embeddings in 3-Ary *n*-Cubes," *Mathematics*, vol. 11, no. 7, p. 1711, Apr. 2023.

[11] X. Liu, J. Meng, and E. Sabir, "Component connectivity of the data center network DCell," *Appl. Math. Comput.*, vol. 444, May 2023, Art. no. 127822.

[12] Z. Chkirbene, R. Hamila, and S. Foufou, "A survey on data center network topologies," in *Proc. Int. Symp. Ubiquitous Netw.* Cham, Switzerland: Springer, 2018, pp. 143–154.

[13] C. Griner, J. Zerwas, A. Blenk, M. Ghobadi, S. Schmid, and C. Avin, "Cerberus: The power of choices in datacenter topology design—A throughput perspective," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 3, pp. 1–33, 2021.

[14] A. C. Castillo, "A comprehensive DCell network topology model for a data center," in *Proc. Int. Conf. Innov. Comput. (ICIC)*, Nov. 2021, pp. 1–6.

[15] Z. Chkirbene, R. Hamila, S. Foufou, S. Kiranyaz, and M. Gabbouj, "Optimization on ports activation towards energy efficient data center networks," in *Proc. Int. Symp. Ubiquitous Netw.* Cham, Switzerland: Springer, 2018, pp. 155–166.

[16] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 123–137, Sep. 2015.

[17] Z. Chkirbene, A. Gouissem, R. Hadjidj, R. Hamila, and S. Foufou, "Exploiting traffic correlation towards energy saving in data centers," in *Proc. IEEE 29th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2018, pp. 1–7.

[18] Y. Zhang, J. Ren, F. Liu, Z. Wang, Y. Song, L. Yin, and Y. Peng, "A novel hyper-converged architecture for power data centers," in *Proc. IEEE Int. Conf. Power, Intell. Comput. Syst. (ICPICS)*, Jul. 2021, pp. 391–394.

[19] P. Cao, S. Zhao, D. Zhang, Z. Liu, M. Xu, M. Y. Teh, Y. Liu, X. Wang, and C. Zhou, "Threshold-based routing-topology co-design for optical data center," *IEEE/ACM Trans. Netw.*, early access, Apr. 13, 2023, doi: 10.1109/TNET.2023.3265276.

[20] F. Hassen and L. Mhamdi, "High-capacity CLOS-network switch for data center networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–7.

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

[22] M. Zhao, Z. Han, and X. Du, "A survey of data center network topology structure," in *Proc. 25th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2023, pp. 303–309.

[23] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *Proc. 9th USENIX Conf. Networked Syst. Design Implement.* Berkeley, CA, USA: USENIX Assoc., 2012, p. 17. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228298.2228322

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

[25] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, and I. Stoica, "A cost comparison of datacenter network architectures," in *Proc. 6th Int. Conf. ACM*, New York, NY, USA, Nov. 2010., pp. 1–16.

[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] *Solutions Small Business Solutions—It Products and Services Cisco Small Business Resource Center Small Business Networking Resources What Are the Different Types of Network Switches?* Accessed: Feb. 14, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/small-business/resource-center/networking/understanding-the-different-types-of-network-switches.html

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 the Tunisia Polytechnic School, in 2012, and the Ph.D. degree in computer science from the University of Burgundy, Dijon, France, in 2017. She was a Research Assistant with Qatar University on a project covering the interconnection networks for massive data centers. She is currently a Postdoctoral Researcher with Qatar University. Her research interests include data center networks, edge computing, green computing, and machine learning as well as deep reinforcement learning techniques.

RIDHA HAMILA (Senior Member, IEEE) received the M.Sc., Lic.Tech. (Hons.), and Dr.Tech. degrees in signal processing and telecommunications from the Tampere University of Technology (TUT), Tampere, Finland, in 1996, 1999, and 2002, respectively. He is currently a Full Professor with the Department of Electrical Engineering, Qatar University, Doha, Qatar. He has been involved in several past and current industrial projects, such as Ooredoo, the Qatar National Research Fund, Finnish Academy projects, and EU research and education programs. He has authored or coauthored over 200 journal articles and conference papers mostly in the peer-reviewed IEEE publications, filed seven U.S. patents, and wrote numerous confidential industrial research reports. His current research interests include mobile and broadband wireless communication systems, mobile edge computing, the Internet of Everything, and machine learning.

ARAFAT AL-DWEIK (Senior Member, IEEE) received the M.S. (summa cum laude) and Ph.D. (magna cum laude) degrees in electrical engineering from Cleveland State University, Cleveland, OH, USA, in 1998 and 2001, respectively. He was with Efficient Channel Coding Inc., Cleveland; the Department of Information Technology, Arab American University, Jenin, Palestine; and the University of Guelph, Guelph, ON, Canada. He is currently with the Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, United Arab Emirates. He is also a Visiting Research Fellow with the School of Electrical, Electronic, and Computer Engineering, Newcastle University, Newcastle upon Tyne, U.K., and a Research Professor with Western University, London, ON, Canada, and the University of Guelph. He has extensive research experience in various areas of wireless communications that include modulation techniques, channel modeling and characterization, synchronization and channel estimation techniques, OFDM technology, error detection and correction techniques, MIMO, and resource allocation for wireless networks. He is a member of Tau Beta Pi and Eta Kappa Nu. He received the UAE Leader-Founder Award, in 2019, the Dubai Award for Sustainable Transportation, in 2016, the Hijjawi Award for Applied Sciences, in 2003, and the Fulbright Alumni Development Grant, in 2003 and 2005. He was awarded the Fulbright Graduate Student Scholarship, from 1997 to 1999. Since 2012, he has been an Editor of IEEE Transactions on Vehicular Technology. He was an Associate Editor of the *IET Communications*, from 2015 to 2020, and *Frontiers in Communications and Networks*, since 2021. He is a registered Professional Engineer in the Province of Ontario, Canada. He is a Distinguished Lecturer of the IEEE (2023–2025).

TAMER KHATTAB (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electronics and communications engineering from Cairo University, Giza, Egypt, and the Ph.D. degree in electrical and computer engineering from The University of British Columbia (UBC), Vancouver, BC, Canada, in 2007. From 1994 to 1999, he was with IBM WTC, Giza, as a Development Team Member and then the Development Team Lead. From 2000 to 2003, he was with Nokia (formerly Alcatel Canada Inc.), Burnaby, BC, Canada, as a Senior Member of the Technical Staff. From 2006 to 2007, he was a Postdoctoral Fellow with The University of British Columbia, where he was involved in prototyping advanced Gbits/sec wireless LAN baseband transceivers. He is currently a Professor with the Electrical Engineering Department, Qatar University, Doha, Qatar. In addition to more than 150 high profile academic journals and conference publications, he has several published and pending patents. He is also a Senior Member of the Technical Staff with the Qatar Mobility Innovation Center (QMIC). He serves as an Editor for the IEEE Communications Letters and an Editor for the IEEE Open Journal of the Communications Society.

Open Access funding provided by 'Qatar National Library' within the CRUI CARE Agreement