IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 28 September 2022, accepted 23 October 2022, date of publication 31 October 2022, date of current version 8 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3218328

RESEARCH ARTICLE

Energy Efficient End Device Aware Solution Through SDN in Edge-Cloud Platform

SUDHANSU SHEKHAR PATRA[1], (Member, IEEE), RAMYA GOVINDARAJ[2], SUBRATA CHOWDHURY[3], (Member, IEEE), MOHD ASIF SHAH[4], RASMITA PATRO[5], AND SUCHISMITA ROUT[5], (Member, IEEE)

[1]School of Computer Applications, KIIT Deemed to be University, Bhubaneswar 751024, India
[2]School of Information Technology and Engineering (SITE), Vellore Institute of Technology, Vellore, Tamil Nadu 632014, India
[3]M.C.A. Department, Sri Venkateswara College of Engineering and Technology (A), Chittoor, Andhra Pradesh 517127, India
[4]Department of Economics, Bakhtar University, Kabul 1001, Afghanistan
[5]School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar 751024, India

Corresponding author: Mohd Asif Shah (ohaasif@bakhtar.edu.af)

ABSTRACT Recently, the networking industries have gone through tremendous changes. It demands high-speed operations and complex problem-solving abilities. To manage these evolutions Internet-of-Things (IoT) is a proposed solution from several technical corners. Numerous researchers and government organizations showing their interest to provide solutions with IoT implementation. Handling a huge amount of network data, its privacy and security, Quality of Service (QoS) requirements and heterogeneity of underlying networking components are the various challenges in IoT implementations. To provide the solution, Software Defined Networking (SDN) is becoming a bliss in managing such complex networking problems. The allocation of the Virtual Machines (VMs) into the end device is an NP-Hard combinatorial optimization problem. We formulate the problem by using simple Additive Weighing (SAW) or Weighted Sum Method (WSM) to allocate the VMs asymmetrically based on CPU Utilization and Memory usage to optimize the energy. The proposed algorithm ServerCons minimizes the number of live migrations and the number of nodes used as well as the energy usage is at par with the state of art algorithms such as First-Fit-Decreasing(FFD), Best-Fit-Decreasing (BFD), and Modified-Best-Fit-Decreasing (MBFD).

INDEX TERMS Cloud computing, end device aware, energy efficiency, IoT, SDN.

I. INTRODUCTION

Over the past few years, the networking and telecommunication field has gone through a series number of changes for providing more smart technologies. Smartness is not limited to technology but it is extended to smart sensing, smart healthcare service, smart home, smart city and smart transportation. There is an evolutionary transformation in every field whether it is machine architecture, operating systems, networking policies, etc. This transformation is possible by combining more than one technology. Software-defined technology is a very important consideration in this field for dealing with the dynamic changing requirements. It gives a new definition to the networking field to deal with dynamic network demands. This facility is not present with traditional networking [1]. The major problem in traditional networking is to incorporate any changes to the networking paradigm major changes are required in network architecture. The static architecture is not flexible enough to adapt to the dynamic network demands. In high traffic volume, the network performance is even worse. The performance parameter further decreases at an exponential rate with growing network size. In all such problems, SDN gives an immense solution to control the network with network intelligence and is considered a boon to the networking industry [2]. These days, the terms ''green computing''and ''green IT'' is widely used to refer to energy-efficient IT solutions. Data center energy efficiency is critical since power and cooling costs account for a considerable portion of their operating costs. Data centers energy-saving techniques not only save money

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

but also contribute to the Green IT objective of reducing carbon emissions. Power consumption in data centers can be reduced in a variety of ways. SDN has been implemented in a wide range of platforms, including institutional and data center networks. Berl et al. [3] proposed the following methods shown in figure 1 for energy optimization that can be applied at various levels of the SDN architecture.

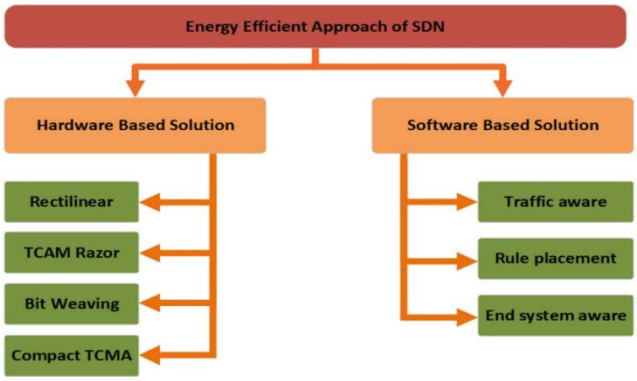

FIGURE 1. Hardware and software-based Energy Saving in SDN.

Hardware improvements are required which makes SDN switches more energy efficient. Such methods aim to reduce the amount of data contained in forwarding switches that needed to be stored in memory. Switch flow makes use of Ternary Content Addressable Memory (TCAM), a particular sort of high-speed memory that can complete a memory search in a single clock cycle. On the controller, software-based methods are used. Software-based Energy optimization strategies for SDN are divided into three categories and are shown in figure 2.

(1) Rule placement (2) End System aware and (3) Traffic aware.

The main contributions of the paper are on the second option i.e., end system aware solution and can be defined as follows:

1. Allocate all the tasks to the VMs.
2. Shutting down the underutilized PMs and running with a smaller number of machines, means that both idle servers, as well as the idle switches, need to be turned off.
3. In data centers, the SDN idea is utilized to create an overlay that links VMs.
4. When VMs and network traffic are consolidated into manageable physical resources in data centers, power consumption is reduced by shutting off idle servers and switches.

A VM is placed on a host by assigning the required resources, such as CPU cores, memory, disc space, and network bandwidth. Because resources are typically over-provisioned, allocating less than required resources can assist condense more VMs and traffic. Figure 3 shows a basic instance of overbooking and consolidation in concept for better understanding which shows the main contribution of the paper, where few PMs can be turned off after consolidating the VMs into fewer PMs. As shown in the figure initially, VMs 1-4 are installed in four hosts and linked over four switches before being overbooked and consolidated. If there is data communication between all four VMs, all four switches should be active and consume power, just like the four hosts. We can observe that the actual use for VM3 and VM4 is much lesser than the allotted capacity. After overbooking, VM3 and VM4 get reduced resources, which may now be consolidated to Host1 and Host2. Following the transfer of VM3 and VM4 to Host1 and Host2, the hosts without VMs, as well as the associated switches, can be turned off.

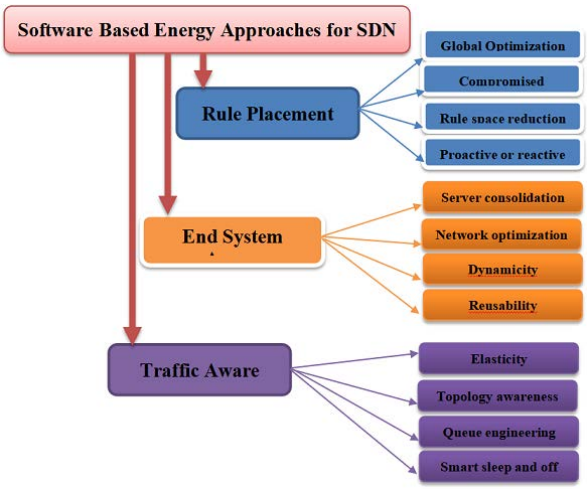

FIGURE 2. Classification of Software-based energy efficiency methods in SDN.

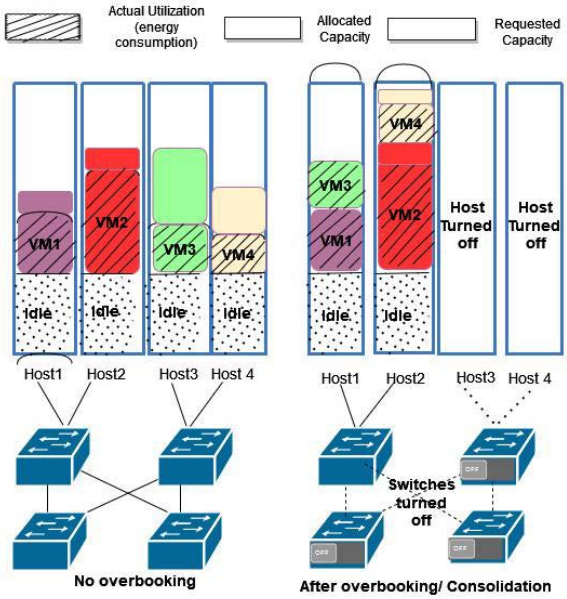

FIGURE 3. Example of consolidation with overbooking.

The rest of the paper is constructed as follows: Section II presents the background, Section III explains the

server consolidation mode for energy optimization and Section IV highlights the experimental setup and analyzes the simulation-related result following a discussion. Finally, Section V concludes the paper with future insights.

II. BACKGROUND AND TERMINOLOGIES

A. ADVENT OF SDN

To offer flexible architecture in handling the edge-cloud interplay [4], [5], recent out-of-the-box technology termed Software Defined Networking (SDN) [6], [7] can be widely used. The network is different from traditional networking. Unlike a traditional network, the router in SDN is decoupled with a data plane and control plane. The separation into two planes makes the network to be robust. The control plane takes all the decisions for the network. It is controlled by a piece of software. The programmable controller is intelligent enough to control the network centrally. It is widely known as a centralized controller. It can take all the packet routing decisions for the network. It is the most worthwhile network choice in the edge-cloud platform. The working principle is decided by the standards set by ONF (Open Networking Foundation). OpenFlow protocol is the best choice to handle the network traffic in SDN. The protocol is smart enough to balance the traffic load and take all the routing decisions. The routing decision is handled by the centralized controller. For every flow of packets, the controller finds the route for the first packet of the flow [8]. The routing table applies to the other packets of this flow. The architecture has three layers: infrastructure layer, control layer, and application layer as shown in figure 4.

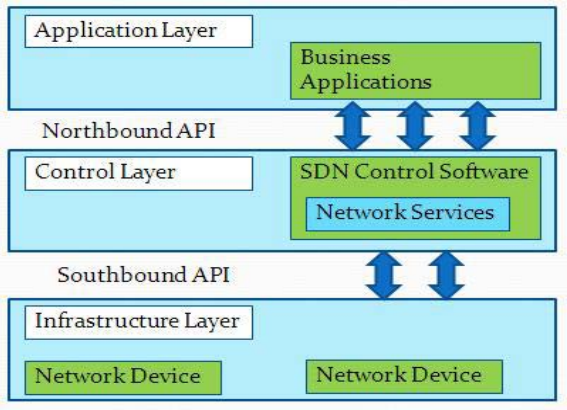

FIGURE 4. Layered architecture in SDN.

Network programmability is not a new concept in ICT. Controlling the overall process of networking is done using the software. SwitchWare [9], [10] provides networking solutions dynamically through software. There are other examples like Click [11], Quagga [12], XOPR [13], and BIRD [14] giving the solutions for programmable routers for routing through the software. These programs are developed in such a way as to modify routing behavior at any time.

Decoupling of control and data plane is not the only feature provided by SDN. It has also been observed in the Routing Control Platform (RCP) [15]. The separation of control and data plane is also earlier observed in the ForCES framework [16], [17], [18]. The centralized controller concept was earlier is also present in Ethane [19], [20], [21]. The separation of the data plane from the control plane adds more scalability and efficiency in the networking industry. It gives high reliability in fault tolerance and software upgradation without affecting underlying packet forwarding. Even in some cases, the router is not divided into two planes, but it supports four plane division. These planes are deciding, dissemination, discovery and data as given by Greenberg in [22].

The specialty of SDN is that it combines the two features of data and control plane separation and network programming under a single roof [23]. SDN provides the mechanism of handling any number of routing or path- finding issues instantly in a very simple way. It does not impose any type of complex networking for finding the same. The separation of control and data plane adds more flexibility for taking any number of complex network decisions. The controlling part is solely taken care of by the control plane. This network intelligence does not require data plane involvement. It enhances the routing ability to make it more focused on the data forwarding path. The switching devices are free from all decisions taking jobs and concentrate only on data forwarding parts. This unique feature of SDN controls the behavior of the network [24], [25].

B. COMPONENTS OF SDN

SDN is an emerging architecture that makes routers free from controlling and routing decision. It separates the network controlling part from the underlying architecture. The underlying architecture seems to have no relation to network programs. This infrastructure part is solely based on the rules of the central controller present at the control layer. It routes the packet based on the flow table prepared by the controller without knowing much detail about it [26]. The controller runs some standardized commands and predefined programming logic to control the network components. ONF in [27] divided SDN into three main functional layers given in figure 4.

1) INFRASTRUCTURE LAYER

This is present as the bottom layer of SDN. This is also known as the data plane of routers. This consists of various network components, forwarding devices, routers and switches used for packet forwarding and switching. All the devices act upon the decision made by the control plane. The routing table is updated as per the control layer, depending on which the packet forwarding decision is made.

2) CONTROL LAYER

This is also known as the control plane. It comprises various controllers that decide to forward the packet through the underlying infrastructure. The controlling decisions comprise software programs based on open APIs. It supervises

the network. It also interacts with four interfaces to control the network: northbound, southbound, eastbound and westbound. It also manages the bandwidth of the network, flow tables, and utilization of the network. The working of this plane is based upon the control logic set by the controller sits in this layer. A network operating system can be used to create the virtual controller with the help of a hypervisor. To control the entire operation a single piece of software information is enough. With the help of a programmable controller, the network can be configured in different environments. With the help of proper flow management principles, the trade-off between energy efficiency and network performance can be optimized. At the same time, the trade-off between energy efficacy and latency is maintained in edge-cloud interplay architecture.

In edge-cloud interplay architecture, the traffic flow is divided into three categories: Active, Wait and Suspended. This helps in proper flow management using the controller. For each flow of packets, the path in the controller is checked. If a path exists, then the flow is added to the waiting queue. When the flow moves to the top of the queue, the flow is active. For the active flow, the routing path is checked. If a path exists, then the flow is routed in that path. Otherwise, the flow is sent to the controller for a new path set up.

3) APPLICATION LAYER

This layer deals with end-user business applications [28]. It is taking care of SDN communication services [29]. It supports a set of applications such as security, QoS, traffic engineering, load balancing, network virtualization, SLA and energy efficiency. SDN controller interacts with the following interfaces.

1. **Northbound interface**: It works in between the control layer and application layer. This is not yet standardized. The REST (Representational State Transfer) [30] API provides an interface for business applications.
2. **Southbound interface**: This interface works in between the control layer and the infrastructure layer to exchange information. NetConf (Network configuration protocol) [31], [32] is an example of this interface.
3. **Eastbound interface**: The interconnection between IP networks and SDN networks is done by the eastbound interface.
4. **Westbound interface**: This interface is taking the responsibility to facilitate the information exchange between various controllers for giving a global view of the network [33].

C. ENERGY CONSUMPTION IN SDN

Energy consumption in SDN is the combined effect of many causes. It is the summation of energy utilization of network device components such as servers, routers, switches and links. Different parameters are involved that cause a major amount of energy expenditure in data centers. It includes the lighting of equipment, UPS power supply, refrigeration and air conditioning, etc [34], [35].

Energy savings can be increased by carefully monitoring energy consumption in all three layers of SDN. It is not possible by working with a single layer only. Energy efficiency techniques must be applied in all layers. End device awareness is based on device awareness. It can save the maximum amount of energy by utilizing the active devices at their maximum level. At the same time, some measures are taken to turn off underutilized devices.

Traffic aware energy efficiency approach is based on taking performance measures varying with traffic conditions. In low traffic, the load of the network is low. By switching OFF maximum devices, energy savings can be done. In high traffic conditions, the load of the controller is more. Different mechanisms are used to alter the control plane action to balance the overall load of the network. It is really a challenging job to adjust the number of active and inactive devices as per the input traffic [36].

Rule placement mechanism deals with energy consumption factors associated with the entry in flow tables. The rules are placed in TCAM. TCAM is very much power-hungry and expensive. The number of rules present in TCAM is directly proportional to the amount of energy consumption. To eliminate this, various routing policies are taken to minimize energy consumption. These policies take care of various constraints like the capacity of switches, links, different protocols, etc. By considering all the parameters it is very difficult to make understandable rules for forwarding the packets and finding optimal solutions for energy savings [37], [38].

III. SERVER CONSOLIDATION MODELLING FOR ENERGY OPTIMIZATION

This article focuses on end device-aware solutions for energy optimization in the data center through SDN. It can save the maximum amount of energy by utilizing the active devices at their maximum level. At the same time, some measures are taken to turn off underutilized devices. Similar to Live migration, VM migration is a pricey process; costs include CPU processing on the migrating node, connection bandwidth between the migrating and migrated nodes, service unavailability on the moving VM, and overall migration time. Therefore, lowering the number of migrations is our top priority. Consolidating the VMs to employ the least amount of nodes is the second objective.

The greatest bound of CPU consumption on a single node must be constrained by a threshold value. To prevent a node's CPU from running at 100% utilization, this is done. This is because a 100% utilization rate may lead to performance loss. The live migration method also consumes some CPU time. A certain amount of CPU throughput must be maintained, hence CPU usage must be kept within that limit. It is crucial to pick the right CPU threshold value since a very high threshold can significantly impact the performance of virtual machines running on a node, while a very low threshold decreases the effectiveness of consolidation. The researchers have not yet decided on the ideal CPU threshold value for

consolidation. Several articles claim that this figure varies between 50% and 75%. Therefore, we choose parameters for our experiment from this range.

In our study, we use two dimensions—CPU and memory—to describe a VM and a node. Let's imagine a physical machine i as a vector with the dimensions Pi (c_i,m_i), where ci represents the CPU capacity in GHz and mi represents the memory capacity in megabytes (MB). Similar to this, VM is represented by the vector Vj (vc_j,vm_j), where vc_j and vm_j stand for the jth VM's CPU and memory capacity, respectively.

Let's say there are M physical machines and K VMs in the fog centre. Let $<s_1,s_2,s_3,\ldots s_M>$ be sets of PMs, and let Si be the PM with the VM identification numbers as $\{i_1,i_2,\ldots i_p\}$. The CPU threshold is represented by the variable T where $T \in [0, 1]$. One must not exceed the limit for the ratio of the total CPU power of the VMs on one PM to this PM's CPU power. The formalization is as follows:

$$\frac{\sum_{j=s_1}^{s_i} vc_j}{c_i} \leq T \quad \forall \textit{ physical machine PM } i \tag{1}$$

The total capacity of a group of VMs in the PM cannot exceed the PM's capacity. The effectiveness of the physical machine consolidation approach can be calculated using three different matrices.

1. The number of PMs that were used
2. The number of PM's that were released
3. The number of VM's that were migrated

A. SERVER_CONSOLIDATION (ServerCons) TECHNIQUE

Some First-Fit and Best-Fit features that try to lower the number of PMs used are inherited by the proposed algorithm. On the other hand, our suggested method decreases the number of migrations as well as the number of PMs required. Algorithm 1 is the proposed algorithm.

This algorithm operates differently from bin-packing techniques. Based on the demand on the VMs, the technique sorts the PMs in decreasing order. The next step is to select a VM for migration, and the weights are ordered in non-increasing order, starting with the last (least loaded) PM on the list. On the first (most busy) PM, we try to allocate them one at a time. If that doesn't work, we move on to the second PM, and so on. This strategy was chosen because the PMs are sparsely loaded at the beginning of the process, thus we want to compact them by moving VMs from the least loaded PMs to the most loaded PMs so that we may release the ones that are least loaded. Up until there are no more alternatives for migration, the stages are repeated. The proposed algorithm and the general bin-packaging algorithms all follows the greedy technique and th ecomparision of them are being described in subsection B.

The CPU and memory capacity of the VMs serve as the two numerical indicators of the PM by VMs. The goal is to put the virtual machines (VMs) inside the PMs to minimize energy consumption because the CPU and memory capacity are asymmetric. Let us consider the CPU and memory loads on a PM i respectively, cl_i and ml_i.

$$cl_i = \frac{\sum_{j=s_1}^{s_i} vc_j}{c_i} \tag{2}$$

$$ml_i = \frac{\sum_{j=s_1}^{s_i} vm_j}{m_i} \tag{3}$$

The PM's can be arranged with only one value to represent the measure of the machine. The measure of a PM can be calculated as:

$$measure(s_i) = \lambda.cl_i + (1-\lambda)ml_i \tag{4}$$

$$\lambda = \frac{\sum_{i=1}^{M} cl_i}{\sum_{i=1}^{M} (cl_i + ml_i)} \tag{5}$$

Similarly, the measures of the VM's depending on their CPU and memory parameters, vc_i and vm_i are:

$$measure(v_i) = \lambda_v.vc_i + (1-\lambda_v)vm_i \tag{6}$$

where λ_v can be calculated as:

$$\lambda_v = \frac{\sum_{i=1}^{Q} vc_i}{\sum_{i=1}^{Q} (vc_i + vm_i)} \tag{7}$$

B. COMPARISION OF ServerCons WITH GENERAL BIN-PACKING ALGORITHMS

Many heuristic techniques exist for the general bin-packing problem including First-Fit (FF), Best-Fit (BF), First-Fit Decreasing (FFD), Best-Fit Decreasing (BFD) and modified Best-Fit Decreasing (MBFD) [39], [40]. We discuss the methods FF, BF and MBFD in Algorithm 2, 3 and 4 respectively to compare them to the proposed ServerCons algorithm. FFD and BFD differ from FF and BF in that objects are sorted by weights in decreasing order before being packed into bins in FFD and BFD. This sorting significantly improves the solution over FF and BF.

An illustration of the proposed algorithm is presented in Appendix 1.

C. ENERGY MODEL

The energy model used is

$$Pow(U) = \begin{cases} Pow_{min} + (Pow_{max} - Pow_{min})XU & if\ U > 0 \\ 0 & otherwise \end{cases}$$

where Pow_{min} is the power consumed by an idle machine (i.e., 70%), Pow_{max} is the power consumed by a fully utilized machine and U is the CPU utilization [41]. The VMs are generated with some random values of dimensions in the given interval [a,b]. For node capacity, we consider two cases:

Algorithm 1 ServerCons

Input: The cluster of Physical machines $S =< S_1, S_2, \ldots.., S_M >$ where $S_i =< i_1, i2, \ldots.i_p >$ are ids of the VMs mapped to machine i; V is the list of VM vector $=< v_1, v_2, \ldots.., v_k >$; N is the node vectors $=< n_1, n_2, \ldots.n_M >$; the configuration of PMs as well as VMs in terms of CPU and memory capacity.
Output: No. of nodes used, total migrations, No. of released nodes

```
while(Migration Attempts Allowed (S)
    for i= 1 to |S|
        Measure[Si] = CalculateMeasure(Si, V, N)
    end for
    S= Arrange nodes by Measure(S,Measure
        index=Find Node index to release(S)
        VMs To Mig = get List of VMs to be
        Migrated(Si)
    Vm_measure[]=CalculateVMMeasure(VMs
    to Migrate)
    VMs to Migrate=Arrange VMs by
        non-increasing order of measure(VMs
        to Migrate, VM_measure)
     m=0
    for each vm ∈ VM to Migrate
        for j=1 to index-1
            Success= Check if Migration is Possible
                (vm, S[j])
            if (success)
                Migrate vm toS[j]
                m=m+1
                break
            end if
        end for
    end foreach
end while
```

End Algo

Function 2 CalculateVMMeasure (VMs to Migrate,V,M)

Input: The CPU and memory capacity of each VMs which node is selected to be released.
Output: The measure of VMs and the parameter λ_v

1. for i=1 to M

do

2. calculate measure(v_i) using Equation (6).
3. Calculate λ_v using Equation (7).
4. return measure(v_i) for all i and λ_v.

End Function

Test cases 1,2,5,6 considered to be equal-sized nodes, while the rest are considered to be variable-sized ones. In test cases 3,4,7,8 the node's CPU and memory are chosen randomly from the values in the braces. The asterisk mark is the mean of the distribution with the highest probability of being chosen, while others are uniformly distributed. For eg., in the

Algorithm 2 First Fit(FF)

1. for i in range (1, m) // m is the number of items
2. do
3. for j in range (1,n) // n is the number of bins
4. do
5. if item i having weight $w_i$accommodates in bin j with capacity c_j then
6. Place the item i in bin j
7. break
8. else
9. continue
10. end if
11. end for
12. if item i doesn't accommodate into any available bin then
13. Make a fresh bin and Place the item into it
14. n++
15. end if
16. end for

Algorithm 3 Best Fit (BF)

1. for i in range (1, m) // m is the number of items
2. do
3. for j in range (1, n) // n is the number of bins
4. do
5. if item i accommodates in the bin j then
6. Compute the remaining availability after the item has been accommodated
7. end if
8. end for
9. Place item i in bin j, where j is the bin with minimum remaining availability after the item is added (i.e., the item "fits best")
10. If such bin doesn't exist, make a fresh bin and Place the item into it
11. end for
12. End Algo

test case 3 $c_i = 6$ is the highest and {4,5,7,8} are uniformly distributed. For each test cases we generate 6 different numbers of VMs: 25,50,75,100,125 and 150. Table 3 shows the experimental results for FFD and ServerCons algorithm for all 8 test cases. We run each case 10 times having each of the nodes the VM numbers are (K=25, 50, 75,100,125,150) and then averaged the results. This gives us more randomness.

D. COMPLEXITY ANALYSIS

The time complexity of the proposed algorithm ServerCons is $O(n^4)$ which is more than the FFD algorithm which is $O(n^2)$. This can be noted that the ServerCons is a complex algorithm as compared to FFD algorithm because ServerCons takes care of two objectives whereas FFD only optimizes one objective. This is negligible when the number of VMs in the data center is less number (maybe approx. 100). When the number of

Algorithm 4 Modified Best Fit Decreasing (MBFD)

Input: PMList, VMList; Output: Allocation of VMs in PMs

```
1. Arrange the VMs in non-increasing order of their uti-
   lization
2. for each VM in VMList
3. do
4.    minimumPower = INFINITE
5.    allocatedPM= NULL
6.    for each PM in PMList
7.    do
8.       if the PM has sufficient resources for VM then
9.          Power=PowerEstimation (PM,VM)
10.         if Power < minimumPower (PM,VM)
11.            alloactedPM= PM
12.            minimumPower =Power
13.   if allocatedPM !=NULL then
14.      Allocate VM to allocatedPM
15. return allocation
```

End Algo

VMs is more in the data center (maybe approx 1000) the higher complexity may be taken into account and a powerful computing node must be allocated to execute this algorithm to reduce the computing time. Even while ServerCons takes longer to run than FFD, it can be justified by one fact. As mentioned earlier live migration is a considerable cost and since the number of live migrations is very less as compared to the other state of art algorithms the increase in running time of ServerCons of $O(n^4)$ is compensated as compared to the running time of $O(n^2)$ and so the server consolidation is very less in ServerCons as compared to FFD. This is negligible when the number of VMs in the data center is less number (maybe approx. 100). When the number of VMs is more in the data center (maybe approx 1000) the higher complexity may be taken into account and a powerful computing node must be allocated to execute this algorithm to reduce the computing time.

IV. RESULTS ANALYSIS

Our proposed algorithm is implemented under Matlab R2014a on an Intel(R) Core(TM) *i*5 − 8250*U* CPU @1.60 GHz 1.80 GHz CPU running on Windows 10 64-bit professional and 8 GB RAM. We have taken 8 test cases as defined in table 1. The comparison of the proposed algorithm with FFD is depicted in table 2.

Figure 5 shows the relation of No. of VMs used Vs No. of migrations. The figure shows as the no. of VMs increases the no. of migrations also increases. In our proposed algorithm ServerCons the number of VMs migrated is less as compared to the other state of art algorithms such as FFD, BFD and MBFD for the same no. of VMs.

In the case of FFD there will be a maximum number of migrations and ServerCons have the least migrations. The difference between the number of migrations is more when the number of used VMs increases. The slope of the line in

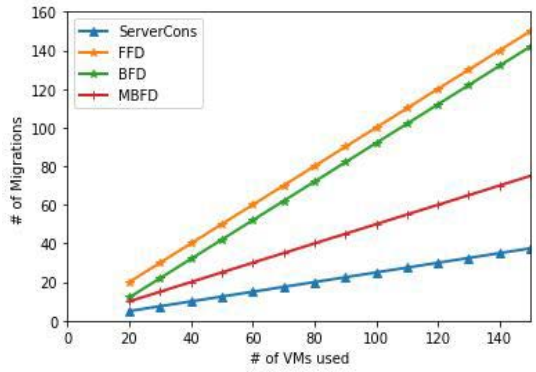

FIGURE 5. Relationship between No. of Migrations and No. of VMs used.

TABLE 1. Test case for the result analysis.

Test Case	Node		VM	
	c_i	m_i	vc_j	vm_j
1	6	4	[1.0,2.0]	[0.5,1.5]
2	10	6	[1.0,2.2]	[0.5,1.5]
3	{4,5,6*,7,8}	{3,4*,5}	[1.0,2.0]	[0.5,1.5]
4	{8,9,10*,11,12}	{5,6*,7}	[1.0,2.2]	[0.5,1.5]
5	6	4	[1.0,2.0]	[vc_j/2,vc_j/2+0.5]
6	10	6	[1.0,2.2]	[vc_j/2,vc_j/2+0.4]
7	{4,5,6*,7,8}	{3,4*,5}	[1.0,2.0]	[vc_j/2,vc_j/2+0.5]
8	{8,9,10*,11,12}	{5,6*,7}	[1.0,2.2]	[vc_j/2,vc_j/2+0.4]

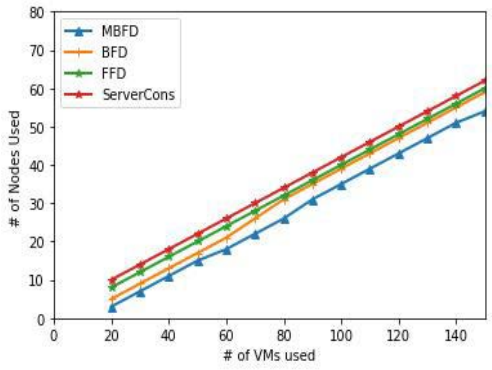

FIGURE 6. Relationship between No. of used VMs and No. of used nodes to accommodate the VMs.

the case of FFD is maximum and our proposed algorithm ServerCons is the least. The result shows that the number of migrations in ServerCons is not only the least but it also grows slowly as compared to the other algorithms. Figure 6 depicts the no. of VMs used Vs. No. of nodes used to accommodate that VMs. The no. of nodes used to accommodate the VMs increases as the no. of VMs increases. ServerCons uses more nodes as compared to the other algorithms FFD, BFD and MBFD because the number of live migrations is less

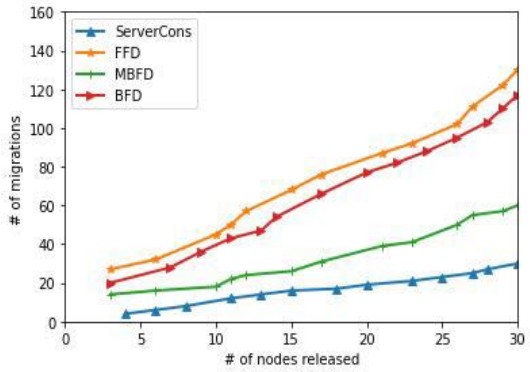

FIGURE 7. Relationship between No. of released nodes and No. of migrations.

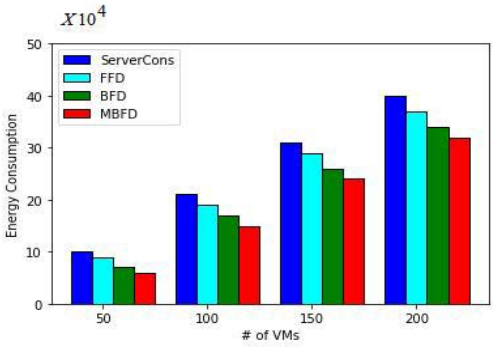

FIGURE 8. Relationship between No. of used VMs and energy consumption.

in comparison to others but is at par comparable to them. Figure 7 depicts the relationship between no. of nodes released Vs. No. of nodes migrated. This figure gives the information for every node released ServerCons needs lesser migrations as compared to the other algorithms. The ServerCons have better performance in comparison to the other traditional algorithms such as FFD, BFD and MBFD. As it can be seen in the graph, ServerCons requires fewer migrations for each node release than other algorithms, and the difference grows as more nodes are released. Figure 8 shows the relation between the no. of VMs used and the energy consumption where the energy consumption with our algorithm is at par in comparison to the FFD, BFD and MBFD algorithms for the same number of VMs used. Though the energy consumption is more as compared to the other algorithms, the number of live migrations is less in comparison to the different algorithms which is a very costly operation.

Figure 9 shows the migration efficiency of the proposed ServerCons and FFD for the test case 5. The migration efficiency is the ratio between the number of released nodes to the number of VMs migrations.

$$\text{Migration efficiency} = \frac{\text{No. of released nodes}}{\text{No. of migrations}} \times 100\%$$

This means if the migration efficiency is 50%, it means that each migration contributing in releasing in 50% of a single node and 100% of migration efficiency tells that each

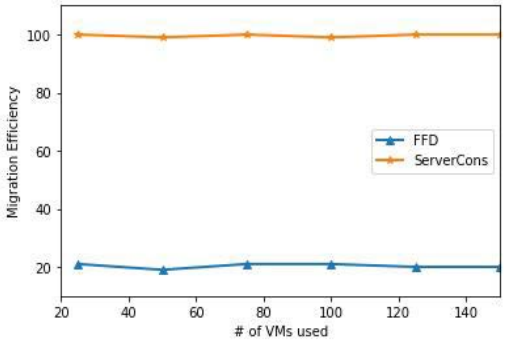

FIGURE 9. Relationship between No. of used VMs and the migration efficiency.

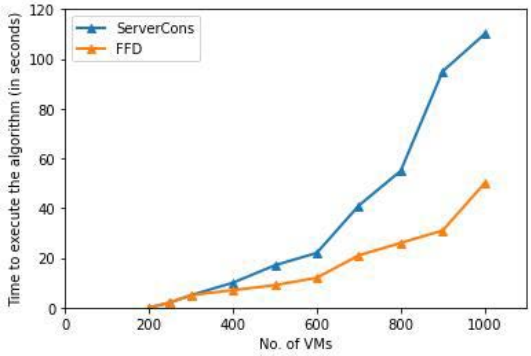

FIGURE 10. Running time of algorithms relative to number of VMs.

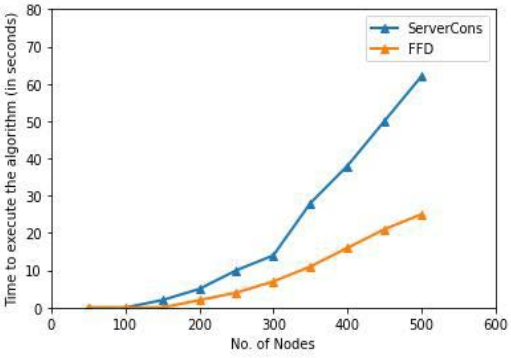

FIGURE 11. Running time of algorithms relative to the number of nodes.

VM migration resulting in releasing one node. The higher the migration efficiency the better the algorithm. Running time of the algorithms FFD, BFD, MBFD and the proposed one ServerCons depends on the no. of VMs as well as no. of nodes. The scalability of ServerCons is investigated by considering the running time. The algorithm is run by taking no.of VMs maximum up to 1000. Figure 10 shows the result of the experiment conducted for FFD and ServerCons algorithms. For ServerCons, it takes roughly 18s to calculate the new placement of 500 VMs, and it takes up to 2 minutes when the number of VMs climbs to 1000. For the FFD algorithm, it takes 10s and 40s, respectively. Figure 11 shows the running time of ServerCons per no. of nodes in the cluster.

TABLE 2. Comparison of ServerCons algorithm with FFD algorithm.

Test Case	No. of VMs	FFD				ServerCons			
		No. of Nodes Used	No. of Nodes released	No. of Migrations	Migration Efficiency	No. of Nodes Used	No. of Nodes released	No.of Migrations	Migration Efficiency
1	25	10.8	4.5	24.2	19%	10.5	4.2	4.3	98%
	50	21.8	9.8	49.4	20%	20.9	8.9	8.9	100%
	75	31.7	13.7	74.2	19%	30.9	12.9	13.1	98%
	100	41.7	19.7	99.3	20%	40.7	18.7	18.7	100%
	125	51	24.1	124	19%	51	24.1	24.5	98%
	150	61.1	28.7	149	19%	61.1	28.7	28.7	100%
2	25	6.4	5.4	23.4	23%	6.7	5.7	7.4	77%
	50	12.4	11.1	48.1	23%	12.3	11	14.4	76%
	75	18.5	15.5	74	21%	18.3	15.3	21.1	73%
	100	24.3	20	98.5	20%	23.9	19.6	27	73%
	125	31	26.8	122.8	22%	30.7	26.5	33.5	79%
	150	36.7	31.3	148.2	21%	36.4	31	40.9	76%
3	25	10.8	5.2	24.5	21%	10.7	5.1	5.1	100%
	50	21.4	9.8	49	20%	20.3	8.7	8.8	99%
	75	31.4	15.1	74.1	20%	30.9	14.6	15	97%
	100	42	21.2	98.5	22%	40.4	19.6	19.7	99%
	125	41.5	25.4	124	20%	51.1	25	25.1	100%
	150	61.3	29.2	148.3	20%	61	28.9	29.2	99%
4	25	6.6	5.9	23.2	25%	6.5	5.8	7.5	77%
	50	12.8	11.7	48.2	24%	12.4	11.3	14.3	79%
	75	19.1	15.1	72.7	21%	18.7	14.7	19.5	75%
	100	24.8	21.2	97.9	22%	24.5	20.9	28.4	74%
	125	30.8	25.4	122.9	21%	30.5	25.1	32.5	77%
	150	36.8	30.7	148	21%	36	29.9	40.3	74%
5	25	10.7	5.1	24.2	21%	10.6	5	5	100%
	50	21	9.3	48.7	19%	20.4	8.7	8.8	99%
	75	31.4	15.5	74.3	21%	30.7	14.8	14.8	100%
	100	41.4	20.3	98.7	21%	40.2	19.1	19.2	99%
	125	51.5	25.1	124	20%	50.8	24.4	24.4	100%

TABLE 2. *(Contiuned.)* **Comparison of ServerCons algorithm with FFD algorithm.**

	150	61.9	30.3	149.2	20%	60.9	29.3	29.3	100%
6	25	6.8	5	22.4	22%	6.8	5	6.7	75%
	50	12.6	10.6	47.6	22%	12.3	10.3	13.7	75%
	75	18.5	15.1	73.9	20%	18.5	15.1	21.1	72%
	100	24.8	20.7	98.6	21%	24.4	20.3	27.8	73%
	125	31.1	25.9	123.5	21%	30.7	25.5	33.4	76%
	150	37.1	31.7	147.6	21%	36.6	31.2	40.9	76%
7	25	10.5	5.1	24	21%	10.1	4.7	4.7	100%
	50	21.2	9.8	49.1	20%	20.1	8.7	8.7	100%
	75	31.6	15.5	73.7	21%	30.2	14.1	14.1	100%
	100	41.2	19.9	98.8	20%	40.2	18.9	19.1	99%
	125	52.3	25.8	124.3	21%	50.6	24.1	24.2	100%
	150	61.2	31.8	149.1	21%	59.7	30.3	30.6	99%
8	25	6.5	5.4	23.3	23%	6.7	5.6	6.9	81%
	50	12.9	10.6	48.8	22%	12.5	10.2	13.7	74%
	75	18.4	14.9	73	20%	18.1	14.6	19.4	75%
	100	25.1	21.5	97.9	22%	24.7	21.1	26.3	80%
	125	30.8	25.8	123.2	21%	30.7	25.1	32.9	76%
	150	36.6	32.5	148.4	22%	36.1	32	42.7	75%

For ServerCons it takes around 60 s to calculate new placement for 500 nodes whereas for FFD it takes 25s.

V. CONCLUSION

In this paper, end device awareness in SDN is implemented. End device awareness is based on device awareness. It helps in saving the maximum amount of energy by utilizing the active devices at their maximum level, minimizes the number of live migrations and at the same time, some measures have been taken to turn off underutilized devices. The experiments are conducted, and the simulation results shown that the proposed algorithm ServerCons is outperformed in comparison to the state of art algorithms in terms of the VM live migration which is a costly operation in the data centers. The energy consumption is at par the state of art algorithms such as FFD, BFD and MBFD. The relationship between the number of VMs used and the migration efficiency has been drawn in which the proposed serverCons migration efficiency is very efficient as compared to the FFD algorithm means every VM migration efficiently releases used nodes. The running time of the algorithm is illustrated and compared with FFD.In the future, we want to find out the single technique that gives more energy saving in both low and high traffic conditions. In low traffic conditions, most of the devices are under low utilization value, whereas, in high traffic conditions, the load of the network is more. Energy consumption of the network is rising with an increase in load value. The behavior of the network components is in two different directions in both the above situations. With the existing network behavior, finding the energy-efficient solution is the most cumbersome task.

APPENDIX

SERVERCONS ALGORITHM ILLUSTRATION

Let the system has M=5 nodes S=<s1,s2,s3,s4,s5> and K=12 VMs <v1,v2,...v12> with the capacities of the VMs in terms of CPU and Memory are shown in the following table 3.

The capacities of PM's in terms of CPU and Memory are {5 GHz,4 GB} each. The CPU threshold is considered to be 0.7 in the problem. The initial stages of the VMs are shown in figure 12 and the measure of each node is calculated.

TABLE 3. Capacities of the VMs in terms of CPU and memory.

	CPU	Memory
VM1	1	0.25
VM2	0.75	1
VM3	0.5	0.75
VM4	1	0.5
VM5	0.5	0.25
VM6	0.5	0.5
VM7	1	1
VM8	1	0.75
VM9	1	0.25
VM10	0.5	0.5
VM11	0.75	0.75
VM12	1	1

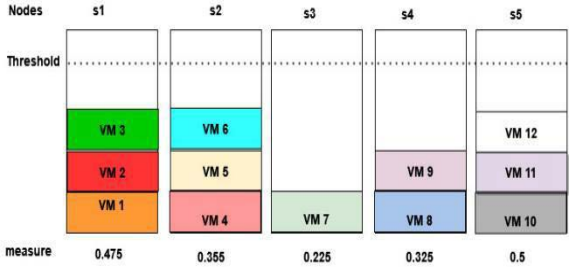

FIGURE 12. Initial stages of PMs with measures.

A. INITIAL STAGE

cl1=(vc1 + vc2 + vc3)/c1 = (1+0.75+0.5)/5= 0.45
ml1=(vm1 + vm2 + vm3)/m1=(0.25+1+0.75)/4= 0.50
cl2=(vc4 + vc5 + vc6)/c2 = (1+0.5+0.5)/5= 0.4
ml2=(vm4 + vm5 + vm6)/m2=(0.5+0.25+0.5)/4= 0.31
cl3=(vc7)/c3 = (1)/5= 0.2
ml3=(vm7)/m3=(1)/4= 0.25
cl4=(vc8 + vc9)/c4 = (1+1)/5= 0.40
ml4=(vm8 + vm9)/m4=(0.75+0.25)/4= 0.25
cl5=(vc10 + vc11 + vc12)/c5 = (0.5+0.75+1)/5= 0.45
ml5=(vm10 + vm11 + vm12)/m5=(0.5+0.75+1)/4= 0.56
λ =(0.45+0.4+0.20+0.40+0.45)/(0.45+0.50)
+(0.4+0.31)+ (0.20+0.25)+(0.40+0.25)+(0.45+0.56)
=0.5
measure (s1)=(0.50)(0.45)+(1-0.5)(0.50)=0.475;
measure (s2)=(0.50)(0.4)+ (1-0.5)(0.31)=0.355;
measure (s3)= (0.50)(0.20)+ (1-0.5)(0.25)=0.225;
measure (s4)= (0.50)(0.4)+ (1-0.5)(0.25)=0.325;
measure (s5)= (0.50)(0.45)+ (1-0.5)(0.56)=0.5.

B. FIRST ITERATION

The server's are arranged based on the measures in decreasing order: <s5,s1,s2,s4,s3>

The last item in the list, that is server s3 is chosen to be released. Server s3 contains a single VM, v7.

measure (v7)= λ_v.vc7 +(1- λ_v).vm7
Here λ_v = 1/(1+1)= 0.5
So, measure (v7)=(0.5)(1)+(1-0.5)(1)= 1

1) COMPUTING MEASURE OF SERVER (s5)

If v7 will be added to server s5:
cl5=(vc10+vc11+vc12+vc7)/c5= (0.5+0.75+1+1)/5=0.65
ml5=(vm10+vm11+vm12+vm7)/m5=(0.5+0.75+1+1)/4=0.81
measure (s5)=(0.5)(0.65)+(1-0.5)(0.81)=0.73 > 0.7
v7 can't be migrated from s3 to s5.

2) Computing Measure Of SERVER (s1)

If v7 will be added to server s1:
cl1=(vc1+vc2+vc3+vc7)/c1= (1+0.75+0.5+1)/5 =0.65
ml1=(vm1+vm2+vm3+vm7)/m1=(0.25+1+0.75+1)/4=0.75
measure (s1)=(0.5)(0.65)+(1-0.5)(0.75)=0.7 <= 0.7
v7 is migrated from node s3 to s1.
The output of the first iteration is shown in figure 13.

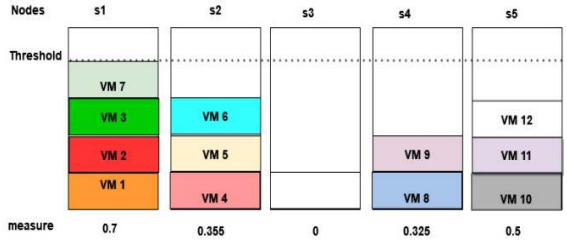

FIGURE 13. Measures of PMs after the first iteration.

C. SECOND ITERATION

The list is stored again in the descending order of their measures as follows:

<s1,s5,s2,s4,s3>

Since s3 doesn't contain any VMs, the next PM selected as the candidate node is s4 and has to be released. S4 contains two VMs, so the measures v8 and v9 have to be calculated.

measure(v8)= λ_v.vc8 +(1- λ_v) . vm8
measure(v9)= λ_v.vc9 +(1- λ_v) . vm9
λ_v = (vc8+ vc9)/ (vc8+ vm8)+ (vc9+ vm9) = (1+1)/(1+0.75)+(1+0.25)= 0.67
measure(v8)=(0.67)(1)+(1-0.67)(0.75)=0.9175
measure(v9)=(0.67)(1)+(1-0.67)(0.25)=0.7525
The VMs in s4 in descending order is <v8,v9>

1) TRY MIGRATING V8 TO s1

Since s1 is fully occupied to 70%, the migration of v8 to s1 is not possible.

2) TRY MIGRATING V8 to s5

cl5=(vc10+vc11+vc12+vc8)/c5 =(0.5+0.75+1+1)/5 =0.65
ml5=(vm10+vm11+vm12+vm8)/m5=(0.5+0.75+1+0.75)/4=0.75
measure (s5)=(0.5)(0.65)+(1-0.5)(0.75)=0.7 <=0.7
So, v8 is migrated to s5.

After migrating v8 to s5, node s5 is fully occupied to the threshold of 70%. So v9 can't be migrated to s5.

3) TRY MIGRATING V9 to s2

cl2=(vc4+vc5+vc6+vc9)/c2 =(1+0.5+0.5+1)/5=0.6

ml2=(vm4+vm5+vm6+vm9)/m2=(0.5+0.25 +0.5+0.25)/4

=0.375

measure (s2)=(0.5)(0.6)+(1-0.5)(0.375)=0.4875 <0.7

So, v9 is migrated to s2.

The output of the second iteration is shown in figure 14.

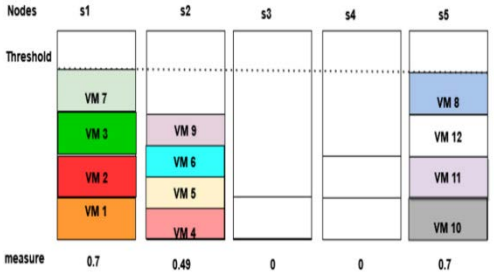

FIGURE 14. Measures of the PMs after the second Iteration.

D. THIRD ITERATION

The list is stored again in the descending order of their measures as follows: <s1,s5,s2,s4,s3>

Since nodes s3 and s4 are empty, node s2 may be selected for migration, but no node will be ready to accommodate the VMs present in s2. So, the migration process stops and the final stage of the nodes is shown in figure 15.

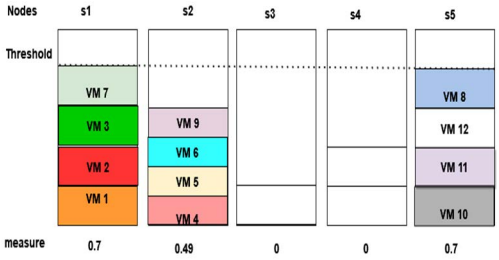

FIGURE 15. Final stage of the PM's where PMs s3 and s4 can be switched off.

REFERENCES

[1] J. Okwuibe, J. Haavisto, I. Kovacevic, E. Harjula, I. Ahmad, J. Islam, and M. Ylianttila, "SDN-enabled resource orchestration for industrial IoT in collaborative edge-cloud networks," *IEEE Access*, vol. 9, pp. 115839–115854, 2021.

[2] P. P. Ray and N. Kumar, "SDN/NFV architectures for edge-cloud oriented IoT: A systematic review," *Comput. Commun.*, vol. 169, pp. 129–153, Mar. 2021.

[3] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis, "Energy-efficient cloud computing," *Comput. J.*, vol. 53, no. 7, pp. 1045–1051, Sep. 2010.

[4] Y. Wu, "Cloud-edge orchestration for the Internet of Things: Architecture and AI-powered data processing," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12792–12805, Aug. 2021.

[5] A. Clarke and R. Steele, "How personal fitness data can be re-used by smart cities," in *Proc. 7th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process.*, Dec. 2011, pp. 395–400.

[6] S. Rout, S. S. Patra, P. Patel, and K. S. Sahoo, "Intelligent load balancing techniques in software defined networks: A systematic review," in *Proc. IEEE Int. Symp. Sustain. Energy, Signal Process. Cyber Secur. (iSSSC)*, Dec. 2020, pp. 1–6.

[7] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by Internet of Things," *Trans. Emerg. Telecommun. Technol.*, vol. 25, no. 1, pp. 81–93, 2014.

[8] J. M. Smith, D. J. Farber, C. A. Gunter, S. M. Nettles, D. C. Feldmeier, and W. D. Sincoskie, "SwitchWare: Accelerating network evolution," Dept. Comput. Inf. Sci., Univ. Pennsylvania, Philadelphia, PA, USA, Tech. Rep. MS-CIS-96-38, 1996.

[9] S. Rout, K. S. Sahoo, S. S. Patra, B. Sahoo, and D. Puthal, "Energy efficiency in software defined networking: A survey," *Social Netw. Comput. Sci.*, vol. 2, no. 4, pp. 1–15, Jul. 2021.

[10] D. S. Alexander, W. A. Arbaugh, M. W. Hicks, P. Kakkar, A. D. Keromytis, J. T. Moore, C. A. Gunter, S. M. Nettles, and J. M. Smith, "The SwitchWare active network architecture," *IEEE Netw.*, vol. 12, no. 3, pp. 29–36, May/Jun. 1998.

[11] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, 2000.

[12] M. R. Nascimento, C. E. Rothenberg, M. R. Salvador, and M. F. Magalhães, "QuagFlow: Partnering quagga with OpenFlow," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 441–442, 2010.

[13] M. Handley, O. Hodson, and E. Kohler, "XORP: An open platform for network research," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 53–57, Jan. 2003.

[14] O. Filip, L. Forst, P. Machek, M. Mares, and O. Zajicek, "BIRD internet routing daemon," NANOG-48, Austin, TX, USA, Tech. Rep., 2010.

[15] L. Yang, R. Dantu, T. Anderson, and R. Gopal, *Forwarding and Control Element Separation (Forces) Framework*, document RFC 3746, 2004.

[16] W. Wang, L. Dong, B. Zhuge, M. Gao, F. Jia, R. Jin, J. Yu, and X. Wu, "Design and implementation of an open programmable router compliant to IETF ForCES specifications," in *Proc. 6th Int. Conf. Netw. (ICN)*, Apr. 2007, p. 82.

[17] T. V. Lakshman, K. K. Sabnani, and T. Y. Woo, "Softrouter separate control network," U.S. Patent 9 014 181, Apr. 21, 2015.

[18] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker, "SANE: A protection architecture for enterprise networks," in *Proc. USENIX Secur. Symp.*, vol. 49, 2006, pp. 137–151.

[19] J. Luo, J. Pettit, M. Casado, J. Lockwood, and N. McKeown, "Prototyping fast, simple, secure switches for etha," in *Proc. 15th Annu. IEEE Symp. High-Perform. Interconnects (HOTI)*, Aug. 2007, pp. 73–82.

[20] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 1–12, 2007.

[21] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Rethinking enterprise network control," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1270–1283, Aug. 2009.

[22] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4D approach to network control and management," *ACM SIGCOMM Comput. Commu. Rev.*, vol. 35, no. 5, pp. 41–54, Oct. 2005.

[23] F. Hao, T. V. Lakshman, S. Mukherjee, and H. Song, "Enhancing dynamic cloud-based services using network virtualization," in *Proc. 1st ACM Workshop Virtualized Infrastruct. Syst. Architectures*, 2009, pp. 37–44.

[24] H. Yan, D. A. Maltz, T. E. Ng, H. Gogineni, H. Zhang, and Z. Cai, "Tesseract: A 4D network control plane," in *Proc. NSDI*, vol. 7, 2007, p. 27.

[25] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, 3rd Quart., 2014.

[26] N. Bizanis and F. A. Kuipers, "SDN and virtualization solutions for the Internet of Things: A survey," *IEEE Access*, vol. 4, pp. 5591–5606, 2016.

[27] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: A survey," *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 24–31, Nov. 2013.

[28] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, and G. Parulkar, "ONOS: Towards an open, distributed SDN OS," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 1–6.

[29] S. Brief, "SDN security considerations in the data center," Open Netw. Found., Palo Alto, CA, USA, Tech. Rep., 2013, pp. 1–12.
[30] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Dept. Inf. Comput. Sci., Univ. California, Irvine, Irvine, CA, USA, 2000.
[31] A. Filali, Z. Mlika, S. Cherkaoui, and A. Kobbane, "Dynamic SDN-based radio access network slicing with deep reinforcement learning for URLLC and eMBB services," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2174–2187, Jul. 2022.
[32] S. D. A. Shah, M. A. Gregory, S. Li, R. D. R. Fontes, and L. Hou, "SDN-based service mobility management in MEC-enabled 5G and beyond vehicular networks," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13425–13442, Aug. 2022.
[33] H. Farhady, H. Lee, and N. Akihiro, "Software-defined networking: A survey," *Comput. Netw.*, vol. 81, pp. 79–95, Apr. 2015.
[34] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and OpenFlow: From concept to implementation," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2181–2206, 4th Quart., 2014.
[35] A. A. El-Saleh, A. Alhammadi, I. Shayea, N. Alsharif, N. M. Alzahrani, O. I. Khalaf, and T. H. H. Aldhyani, "Measuring and assessing performance of mobile broadband networks and future 5G trends," *Sustainability*, vol. 14, no. 2, p. 829, Jan. 2022.
[36] I. Bermudez, S. Traverso, M. Mellia, and M. Munafo, "Exploring the cloud from passive measurements: The Amazon AWS case," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 230–234.
[37] K. R. Jackson, K. Muriki, L. Ramakrishnan, K. J. Runge, and R. C. Thomas, "Performance and cost analysis of the supernova factory on the Amazon AWS cloud," *Sci. Program.*, vol. 19, nos. 2–3, pp. 107–119, 2011.
[38] B. Wilder, *Cloud Architecture Patterns: Using Microsoft Azure*. Sebastopol, CA, USA: O'Reilly Media, 2012.
[39] S. Fang, Y. Yu, C. H. Foh, and K. M. M. Aung, "A loss-free multi-pathing solution for data center network using software-defined networking approach," in *APMRC Dig.*, 2012, pp. 1–8.
[40] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella, "Toward software-defined middlebox networking," in *Proc. 11th ACM Workshop Hot Topics Netw.*, 2012, pp. 7–12.
[41] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. NSDI*, 2010, vol. 10, no. 8, pp. 89–92.

SUDHANSU SHEKHAR PATRA (Member, IEEE) received the M.C.A. degree in computer applications from the Motilal Nehru National Institute of Technology, Allahabad, India, in 1997, the M.Tech. degree in computer science and engineering from Utkal University, Bhubaneswar, India, in 2008, and the Ph.D. degree in computer science from KIIT University, Bhubaneswar, in 2013. He is currently working as an Associate Professor with the School of Computer Applications, KIIT Deemed to be University, Bhubaneswar. He has worked in IT Industry for more than six years in India and Berlin, Germany. His research interests include grid computing, cloud computing, algorithms, SDN, and machine learning. He guided two Ph.D. students and currently guiding five Ph.D. students. He has around 60 research papers in various journals and conferences, 15 book chapters. He is a member of ISTE.

RAMYA GOVINDARAJ received the B.Tech. degree in IT from the Adhiparasakthi College of Engineering–Anna University, India, in 2006, the M.Tech. degree in IT from VIT University, India, in 2009, and the Ph.D. degree from VIT Vellore, in 2020. She has published more than 30 articles in international journals 20 papers in international conference. She has published books, book chapters, and patented products rights for her works. She has conducted many internationals conference FDP, Workshops, and Seminars. She is currently working as an Assistant Professor (Senior) with the School of Information Technology, VIT, Vellore, India. She been the committee members of many international journals and Science Societies.

SUBRATA CHOWDHURY (Member, IEEE) is currently working as an Assistant Professor with the Department of CSE, Sri Venkateshwara Engineering College, Andhra Pradesh, India. He has worked in the IT Industry for more than five years in the research and development, he has handled many projects in the industry with much dedications and perfect time limits. He has been handling projects related to AI, blockchains, and the cloud computing for the companies from various national and internationals clients. He had published four books, from 2014 to 2019, at the domestic market and Internationally Publishers like CRC, River. He has been the editor for the two books for the CRC and River publisher. He has published more than 30 papers and copyrights and patents in his names. He has participated in the Organizing Committee, Technical Programmed Committee, and Guest Speaker for more than ten conferences and the webinars. He also reviewed and evaluated more than 50 papers from the conferences and the journals book chapters and science articles in AI, data science, the IoT, blockchain, and cloud computing for CRC, Springer, Elsevier, Emerald, IGI-Global, and InderScience Publishers.

He has been awarded by the International and Nationals Science Societies for his eminence contributions in the research and development field. He has received travel grants and also member of the IET, ISTE, ACM, and other Accretional bodies. He is the Associate Editor for the *JOE* IET, Wiley, and other Journals. He has taken parts in the workshops, webinars, FDPs as the Resource persons.

MOHD ASIF SHAH is currently working as an Associate Professor with Bakhtar University, Kabul, Afghanistan. He has been earlier working as an Assistant Professor of economics with the Forbes Business School, India, and LPU, India. He has also served as a Lecturer at the JCE, Jammu and Kashmir, India, and also helped his department with teaching assistance during the Ph.D. He has published more than twenty research papers (SCI/WOS/UGC Indexed) with more than thirty citations.

RASMITA PATRO received B.C.A. and M.C.A. degrees from Indira Gandhi National Open University, New Delhi, India. She is currently pursuing the Ph.D. degree with the KIIT Deemed to be University Bhubaneswar, India. She has published three papers in international conferences. Her research interest includes cloud and fog computing.

SUCHISMITA ROUT (Member, IEEE) received the M.Tech. degree in computer science from the National Institute of Technology Rourkela and the Ph.D. degree in computer science from KIIT University. She is currently working as an Associate Professor with the School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar, India. Her research interests include SDN, mobile *Ad hoc* networks, cloud computing, edge computing, and the IoT. She has more than 22 research papers in various international journals and conferences of repute. She is a Active Member of ISTE and CSI.