Energy aware adaptive resource management model for software-defined networking-based service provider networks

Mahmud Rasih Çelenlioğlu¹ | Haci Ali Mantar¹²

¹Department of Computer Engineering, Gebze Technical University, Kocaeli, Turkey
²Department of Electrical and Computer Science, Syracuse University, Syracuse, New York, USA

Correspondence
Mahmud Rasih Çelenlioğlu, Department of Computer Engineering, Gebze Technical University, Kocaeli 41400, Turkey.
Email: mcelenlioglu@gtu.edu.tr

Funding information
The Scientific and Technological Research Council of Turkey (TUBITAK), Grant/Award Number: 113E253

Abstract
Service provider networks (SPNs) seek to implement the resource management mechanisms called load balancing and energy saving to satisfy today's networking demands. Software-defined networking (SDN) allows SPNs to perform agile and efficient networking operations. It is still an open issue on how to implement these resource management mechanisms in SDN-based SPNs. An energy-aware adaptive resource management model (ERM) for SDN-based SPNs is proposed. Load balancing aims to distribute overall network traffic volume as fairly as possible among available links. Energy-saving aims to turn off as many network devices and links as possible. Unlike the works in the literature, our model utilises the trade-off between load balancing and energy efficiency. To do so, the controller establishes paths among edge pairs. The controller aggregates traffic load between edge pairs to as few links as possible based on a predefined load level. Then, unused network elements switch to sleep mode. In the case of load balancing, the controller distributes traffic load among active network resources. Experimental results show that ERM can successfully increase both energy-saving and resource utilization even under heavy traffic loads.

1 INTRODUCTION

Advancements in both wired and wireless networks, the proliferation of mobile devices and applications have changed the way we use the Internet usage. Within the last 2 decades, there have appeared applications and services such as e-commerce, on-line gaming, social media, cloud services, and the Internet of Things (IoT). Accordingly, the volume of Internet traffic has increased, and Internet traffic characteristic has differentiated dynamically [1, 2]. Due to the lack of smart and autonomous systems, service providers (SPs) usually serve their network resources such as bandwidth in full capacity to satisfy diverse user demands and accommodate traffic bursts. Thus, network devices such as routers and switches are active all the time, even if network traffic is relatively low compared to peak times [3]. This results in 2%–7% of the World's electricity consumption in Information and Communication Technologies (ICTs) [4, 5]. Besides, the energy consumption of ICTs has been growing 3% every year [6]. Thus, SPs need smart network management systems that not only increase resource utilization but also decrease energy consumption.

Software-defined networking (SDN) [7] is a new networking paradigm that emerged to satisfy today's networking demands. It separates control and data planes to provide centralised management, virtualisation, agility, programmability, and efficiency. These abilities allure the attention of SPs since they aim to increase resource utilization, decrease operational costs, and protect the environment. SDN-based resource management models proposed in the literature either perform traffic load balancing to increase resource utilization or energy-saving both to decrease costs and protect the environment. The works that focus on load balancing distribute the network traffic as fairly as possible among links. Others that focus on energy saving aggregate network traffic to put as many links and switches as possible into sleep mode. Therefore, these two operations are usually considered as opposite. To the best of our knowledge, our work is the first work that focuses on utilising the trade-off between these two resource management mech-
anisms in SDN-based networks. Apart from this, the SDN controller may become a bottleneck in terms of decision making and signalling as the network size, and the number of flows increases. It is still an open issue on how to perform these operations scalably in SDN-based Service Provider Networks (SPNs).

In this work, we propose an energy-aware adaptive resource management model (ERM) that utilises the trade-off between energy saving and load balancing. The proposed model is constructed upon our previous works [8, 9]. Briefly, pre-established multi-paths between edge ( ingress-egress) nodes abstract the physical complex network into a simple virtual one. Upon that, the controller performs routing, admission control, signalling, and load balancing in a scalable manner by taking paths into account. Apart from the works in the literature, we focus on utilising the trade-off between energy-saving and load balancing in the proposed model. To the best of our knowledge, this is the first work in literature that energy-saving and load balancing operations work in harmony. To make this happen, we introduce a trade-off value that determines the importance of these two operations against each other. If trade-off value favours energy-saving, the controller aggregates edge traffic into fewer paths and puts unutilised network elements such as links and switches into sleep mode. If the trade-off value favours load balancing, the controller saves less energy and focuses on distributing the traffic among paths. In line with the explanations above, we summarise the contributions of the paper below.

- To the best of our knowledge, this is the first work in the literature that balances the trade-off between load balancing and energy saving. Depending on the traffic volume of the network, the network administrator can adjust the level of energy saving.
- Adaptive energy saving and load balancing mechanisms are proposed to provide scalable resource management.
- The novel load balancing mechanism works in harmony with the proposed energy-saving mechanism. To make this happen, the controller balances traffic of edge pairs among active paths.
- The controller can communicate with data plane elements over the modified OpenFlow [10] protocol to sleep and wake up them.
- The controller performs routing, admission control, signalling, and resource management operations in a scalable manner exploiting pre-established multi-paths.
- The proposed model can be applied to non-SDN-based SPNs by replacing switches on the edge with SDN-capable ones.

The rest of this work is organised as follows. Section 2 discusses related works. Section 3 describes background information about the proposed model. Section 4 presents design and implementation details of the proposed model. Section 5 presents the experimental results and the evaluation. Finally, the work is concluded in Section 6.

## 2 RELATED WORKS

This section briefly explains the related works regarding to energy saving in SDN-based networks. Basically, there are two approaches that have been adopted for energy efficiency in SDN-based networks. These are link rate adaptation (LRAA) and power-down approaches. In the first approach, the link rate is adapted according to the traffic load. In the power down approach, ports, line cards, and integrated chassis of routers and switches are either turned off or put into sleep mode. The LRAA approach contributes less energy saving compared to the power down approach [11, 12]. However, the power down approach causes routing oscillation and delay [13]. There are also some works that exploit both approaches.

The idea in Reference [14] is to reroute flows on existing paths to adjust link loads in a way that LRAA-based energy saving is maximised. In the scope of this study, the idea mentioned above is defined as a Mixed Linear Integer Problem (MILP). Then, three greedy algorithms and one genetic algorithm-based heuristic algorithm (GA) are proposed for redirecting flows over existing paths. Experimental results show that GA outperforms greedy algorithms. In this work, several paths are computed per pair before rerouting, whenever the controller executes proposed algorithms. Thus, the control plane and signalling scalability are limited. Although two real networks are used for assessment of the proposed algorithms, real traffic trace is not used. In addition, there are also no admission control and resource management in terms of load balancing mechanisms.

The following works utilise PDA to perform energy saving. The work [15] provides a 0-1 Integer Linear Programing (ILP) model that maximises energy saving in a global manner by taking energy consumption of integrated chassis, line cards, and ports of routers into account. It proposes two greedy algorithms namely Alternative Greedy Algorithm (AGA) and Global Greedy Algorithm (GGA) for energy saving. These two algorithms mainly reroute flows on different paths when the network has a relatively low traffic load. This work has routing and signalling scalability issues. The controller performs rerouting per flow. Besides, it communicates with all the switches along the new path per flow.

The work [16] combines IEEE802.az, that is energy efficient Ethernet and SDN-based Segment Routing (SR) to save energy. SDN controller computes link costs based on two metrics, which are EAGER and CARE. The EAGER metric takes link utilization into account. The CARE metric, on the other hand, takes both link utilization and congestion threshold into account. Two energy saving algorithms proposed in the work are tunnelling TE (TTE) and Non-tunnelling TE (NTE), respectively. In TTE, the controller performs path computation per source and destination pairs in the first phase. Then, the controller establishes SR tunnels. In the final phase, the controller puts unused links into sleep mode. In NTE, the controller computes paths and decides links that will be put into sleep mode. In the second phase, there is no tunnel establishment. The controller only puts the designated links into sleep mode. Finally, controller computes...
ECMP routes between source and destination pairs using awake links. This work in fact has a similar idea that we proposed in our preliminary work [9]. Routing and signalling scalability are achieved via pre-determined paths. However, there is no contribution regarding admission control. Although it is said that load sharing is performed, authors do not mention how this is achieved.

Software Defined Green Traffic Engineering (SDGTE) [17] framework minimises active links and switches in backbone networks without knowing the future traffic demand. Authors provide an ILP definition of energy efficient routing problem. Whenever a flow arrives, the controller performs ILP based routing computation. Apart from energy-efficient routing, SDGTE reroutes flows both in under-utilised and over-utilised links to minimise energy consumption further and to reduce link utilization. In under-utilization case, links whose utilization is below the predefined threshold are determined, and flows on those links are rerouted. Then, the controller puts these links into sleep mode. In the latter case, the controller identifies the over-utilised links and re-routes the flows on those links to reduce link utilization. Apparently, the controller performs routing computation per flow whenever a new flow comes. Besides, the controller also performs re-routing in case of over and under link utilization.

Authors in [18] propose a multi-objective routing approach in a multi-controller control plane. To achieve these objectives, a multi objective evolutionary algorithm, called SPEA2 is developed. SPEA2 performs routing in a way that energy consumption and traffic delay is minimised without degrading the performance. Meanwhile, it takes both controller to switch and switch to switch loads into account. Whenever the controller receives a flow request, SPEA2 calculates a path taking data and signalling load into account. Subsequently, the controller establishes a path for the requested flow. The control plane in this work has multiple controllers. Thus, it satisfies routing and signalling scalability at some level.

Green Application Layer is an ETSI standard [19], which is a framework for exchanging information between control and data planes. The work [20] integrates GAL with SDN to exchange information regarding power management of data plane entities with a controller. This model adopts both PDA and LRAA. In that sense, it proposes an ILP model for the allocation of resources optimally based on network load and actions of flow tables. Experimental network in this study is small. The controller may become the bottleneck as network size, number of flows, and actions in flow tables of switches increase. Besides, the signalling scalability issue appears as flows are rerouted.

Authors in [21] aims to minimise active links and adapt discrete link rates to traffic load for saving energy. Firstly, they provide a mixed-integer programing definition of the problem. Secondly, they propose a heuristic algorithm, which identifies most energy consuming flows and reroutes them on alternative paths for reducing energy consumption. First, the proposed algorithm computes the shortest path for each flow. Subsequently, it calculates the energy consumption of the whole network. The value found at this stage is the upper bound for energy consumption. In the third stage, the proposed algorithm removes each flow one by one to identify its impact on energy consumption. To do so, a weighted graph is generated for each flow. Finally, the controller computes k-shortest paths and the with the least energy consumption is selected as a route. Obviously, the scheme has routing and signalling scalability issues since the controller performs per flow routing and rule installation.

Researchers in [22] propose two algorithms for allocation of Virtual SDN (VSDN) in a reliable and energy efficient manner. Relative Disjoint Path (RDP) generates two trees based on a redundancy factor. Then it merges the two trees to obtain a graph where links and nodes exist in both. In this way, there is only one path if the redundancy factor is 0. Similarly, there are two disjoint paths if the redundancy factor is 1. Otherwise, one or more links are shared among two paths.

State-Aware of Bandwidth and Energy Efficiency (SA-BEE) algorithm allocates VSDNs based on available bandwidth and energy consumption factor. It adaptively increases or decreases energy consumption factor based on the network state at first. Then it generates a weighted graph for a source node to all other nodes. Finally, it looks for lower energy consuming paths between the source node and all other nodes.

In GreSDN [23], the controller performs energy-efficient routing without re-routing of flows. It maintains two topologies. The first topology is the physical topology, which contains all the links and switches. The second topology is the virtual topology, which contains only awake links. The controller performs per-flow routing bearing the two topologies in mind. If there is an inactivate link along the path, the controller sends a signal to the corresponding switch to awake the link. Meanwhile, the device management module within the controller periodically checks the state on links. If a link is not used, it puts the link into sleep mode. The authors propose two routing algorithms; Constant Weight Greedy Algorithm (CWGA) and Dynamic Weight Greedy Algorithm (DWGA). Both algorithms perform path computation on two graphs. These graphs are generated per request. At the time of graph generation, the controller removes any link where the summation of the current load with requested bandwidth exceeds its capacity on both physical and virtual graphs. Finally, CWGA and DWGA perform routing computation based on static and dynamic link costs, respectively. Apparently, the controller performs per-flow routing. Thus, it suffers from a severe routing scalability issue. Besides, the controller must communicate with all the switches along the path during the path establishment process. This also results in the signalling scalability issue.

In the work [24], authors propose a heuristic algorithm (ETALSA) for energy saving via only powering up/down links. ETALSA that runs on a controller takes energy prices into account apart from other works. During the execution of ETALSA, the controller iteratively selects switches from the one with the highest energy cost to the lowest. Then, it computes the utility value per link, which is connected to the selected switch. The utility value is computed based on the connectivity of switches, traffic demand, and energy prices.
Finally, the controller powers a link down based on the utility value if there becomes less energy consumption. This work resembles our preliminary work [9] in a way that pre-established multi-paths exist both for scalability and resource management. The main differences are that the work [24] powers only the links down and it takes energy prices into account.

Consequently, the works in the literature mostly perform either load balancing or energy saving. Our work differentiates itself from the works in the literature, performing energy saving in compliance with load balancing.

3 | BACKGROUND

ERM relies on pre-established paths (PEPs) that connect ingress and egress nodes. The controller performs resource management operations, which are energy-saving, load balancing and path capacity updates, based on PEPs. Routing, admission control, and signalling scalability are also achieved in favour of PEPs as presented in our previous work [8]. Apart from the previous work, we propose ERM for SDN-based SPNs. The proposed model exploits the trade-off between energy-saving and load balancing. We propose a novel energy-saving algorithm that is adaptive to the traffic load. Additionally, previously proposed load balancing and resizing algorithms are revised so that controller performs these operations in harmony with the energy-saving operation.

For the purpose of clarification, Tables 1 and 2 summarise main acronyms and notations used in this work, respectively.

3.1 | Architectural overview

The proposed model is illustrated in Figure 1. The data plane resides at the bottom and consists of provider edge and core switches. Ingress SDN switch (ISS) and Egress SDN switch (ESS) are the switches where traffic enters and leaves the network, respectively. An edge switch is both ISS and ESS at the same time because traffic is bidirectional. In addition to edge switches, there are core switches, which we name Core SDN switches (CSSes). A set of CSS connects edge switches and solely perform packet forwarding. In the upper layer, the proposed controller resides. It is logically centralised and mainly responsible for resource management (i.e. load balancing, active capacity resizing, and energy saving), routing, and admission control operations. To do so, the controller abstracts the single physical network into multiple virtual networks using PEPs between edge switches. All operations are performed based on these paths. The two layers in Figure 1 communicate with each other via a signalling protocol (e.g. OpenFlow [10]).

The proposed model has PEPs between each edge pair (EP), such as ISS₁–ESS₁ in Figure 1. These paths are virtual and simplify the complex physical network. A simpler virtual network consists of edge switches and PEPs. The controller performs resource management operations, routing, admission control, and signalling based on the virtual network. Briefly, controller aggregates flow into less number of paths adaptive to the traffic load and then puts unused links and switches into sleep mode for energy saving. It also equalises the cost of paths that belong to the same EP to achieve load balancing. Controller, additionally, adjusts the capacity size of PEPs concerning traffic load to improve resource utilization more.

Notice that the controller does not perform routing and admission control operations per flow. In case of routing, the

| TABLE 1 Summary of acronyms |
|------------------------------|
| **Acronym** | **Explanation** |
| ERM | Energy aware resource management model |
| RASO | Routing, admission control and signalling operations |
| SP | Service provider |
| SPN | Service provider network |
| PEP | Pre-established multi-path |
| ISS | Ingress SDN switch |
| ESS | Egress SDN switch |
| CSS | Core SDN switch |
| EP | Edge pair |
| LSDB | Link state database |
| PSDB | Path state database |
| RACM | Routing and admission control module |
| LBM | Load balancing module |
| PRM | Path resizing module |
| ESM | Energy saving module |
| PPM | Path management module |
| ICM | Information collection module |
| ICCM | Inter controller communication module |
| EPMA | Energy aware path management algorithm |
| HMA | Hash-based Modulo-ρ assignment operation |

| TABLE 2 Summary of notations used in this study |
|-----------------------------------------------|
| **Notation** | **Meaning** |
| V | Set of switches |
| E | Set of links |
| \( P^{sd} \) | Set of paths between edge pair \((V_s, V_d)\) |
| \( P^{ik} \) | A path belongs to \( P^{sd} \) |
| \( E_i \) | A single physical link \( i \) |
| \( L_{x}^{ik} \) | Virtual link of \( P^{ik} \) on physical link \( E_i \) |
| \( \rho(...) \) | Capacity of pair, path or link |
| \( \theta(...) \) | Load of pair, path or link |
| \( \lambda(...) \) | Cost of pair, path or link |
| \( \varepsilon(...) \) | Energy of pair, path or link |
controller simply assigns a flow to a path. In the case of admission control, the controller decides based on the state of paths, which are stored in both link state database (LSDB) and path state database (PSDB). PEPs also allow the controller to manage the whole network with less signalling. This is because the controller does not communicate with each node along the path per flow in routing, admission control and resource management operations. It only communicates with edge switches, instead. Therefore, scalable routing, admission control and signalling are achieved in favour of PEPs.

3.2 Network abstraction

The physical network is represented by a graph $G = (V, E)$. The set $E$ contains links indexed by $i$. The set $V$, on the other hand, contains all nodes (i.e. ISS, ESS, CSS). We denote an ISS-ESS pair as $(V_n, V_d)$ where $V_n \neq V_d$. Note that pair $(V_n, V_d)$ is the same pair in reverse direction as $(V_d, V_n)$ since any edge node is both ISS and ESS at the same time. The controller converts the physical network to a virtual one. Then, it performs operations based on the virtual network. The virtual network consists of set $E^{vd}$ and $P^{vd}$ of virtual links and PEPs per $(V_n, V_d)$, respectively. We denote a path $k$ that belongs to a pair $(V_n, V_d)$ as $P^k_{vd}$. Any two different paths $P^k_{vd}$ and $P^l_{vd}$ of a pair $(V_n, V_d)$ is mutually disjoint since a path $P^k_{vd} \subseteq 2^E$. Cardinality of paths per pair $(|P^{vd}|)$ must be at least two due to multi-path and can be as many as network topology allows. Besides, number of paths for each pair can be different.

Network administrator makes the decision of which path has how many PEPs. In the scope of this work, we assume that PEPs are already exist in the network. Paths between EPs can be established using Virtual LAN (VLAN), MPLS or SR. In compliance with this, we do not develop any multi-path computation algorithm as there are already several algorithms used in the literature, such as $[25–27]$. $|P^{vd}|$ per $(V_n, V_d)$ is defined by network administrator. Note that PEPs can be established regarding to any cost value such as residual bandwidth, latency, etc.

Notice that any two paths where each of which belongs to a different pair may share one or more physical links. We name such paths and pairs as neighbour paths and pairs, respectively. For the sake of simplicity, the controller virtualises links based on the number of neighbour paths per link. We denote a virtual link which serves to a path $P^k_{vd}$ as $L^k_{vd}$. Capacity of a physical link, denoted as $\varphi(L^k_{vd})$, is portioned out to its virtual links based on the load of neighbour paths. In compliance with this, a path capacity $\varphi(P^k_{vd})$ becomes the capacity of its virtual link which has the minimum capacity ($\min(\varphi(P^k_{vd}))$). For example, there are two paths on the physical link $E_0$ in Figure 1. Hence, its virtual links are $E_0^{1,1}$ and $E_0^{1,2}$, where $E_0^{1,1}$ serves to the path $P^{1,1}$ of pair ISS$_1$ - ESS$_1$ and $E_0^{1,2}$ serves to the path $P^{1,2}$ of pair ISS$_2$ - ESS$_1$.

3.3 Controller design

The controller has two databases, four routing, admission control and resource management modules and three helper modules. The two databases are maintained in the background without interfering operations performed by modules. Routing, admission control and resource management modules have main tasks to handle but they can cooperate with other helper modules.

LSDB keeps track of physical network elements that are links and switches. Some of them are identifier of $E_i$ corresponding switch datapath identifiers (DPID) and port numbers that $E_i$ connects, maximum transfer rate, power consumption and link load. Note that link load is not a static value unlike others because it changes in time regarding to the network load. PSDB to keep track of PEPs and other information regarding to them. This database is the virtualized version of the physical network. Controller mostly use this database for network management purposes. PDSD contains both static and temporary data. Some of the static information are path identifier (PID), datapath identifiers of edge nodes (i.e. ISS and ESS), links that form the path. Some of the dynamic data are path capacity and path load.

Routing and admission control module (RACM) allows flows to be routed over the associated edge nodes. In routing, the controller assigns the incoming flow to a path of the corresponding pair. In admission control, the controller checks available resources before making a decision regarding the routing. This way, it prevents the overloading of resources. If there is an available resource, the controller performs routing of the flow and follows the standard routing procedure.
Otherwise, it simply rejects the flow. Load balancing module (LBM) basically equalizes the cost of paths per pair. At first, it computes the path costs of the corresponding pair and calculates the equalization cost. Afterwards, it shifts flow in overloaded paths to underloaded paths. Path resizing module (PRM) updates the capacity of paths depending on loads of neighbor paths and pairs. Whenever load of path exceeds a certain threshold, the resizing procedure is invoked. Energy saving module (ESM) aggregates information from less number of paths. Then, it puts network elements that are not needed into sleep mode.

Path management module (PMM) establishes, updates or removes paths PEPs. Information collection module (ICM) collects data from underlying switches and performs necessary calculations on them to extract the information required for routing, admission control, load balancing, capacity resizing, and energy-aware resource management. The controller maintains all the information in databases. Inter controller communication module (ICCM) is responsible for communicating with other controllers in the control plane. This module prevents the network from a single point of failure problem. When the single controller stops working, all the routing, admission control, and resource management operations stop. In this work, we do not address this issue as there are proposed works mentioned in [28] regarding this. A simple solution for our model is that there can be two or more controllers within the control plane. One of them would be the master, and the other controllers, a.k.a. backup controllers, periodically check if the master controller is alive. In case of failure of the master controller, one of the backup controllers takes over. As the proposed model maintains network state in databases, the backup controller can directly connect to LSDB and PSDB to manage the whole network. Thus, there is a minimal overhead of taking control in the case of controller failure.

4 | ENERGY-AWARE ADAPTIVE RESOURCE MANAGEMENT

4.1 | Energy saving

There occurs a trade-off between optimal energy saving and computation overhead. In today’s highly dynamic Internet traffic, the controller may become a bottleneck while computing optimal energy saving. Additionally, the optimal energy state may not last long due to the traffic fluctuations. Thus, we prioritize scalability over optimal energy saving. To do so, our controller makes EP-based energy-saving computation. Upon completion of pair-based optimization, the controller aggregates pair traffic into designated paths. Subsequently, it deactivates idle links and switches in the whole network. This local energy saving converges to a global energy saving in time.

To be clearer about how energy can be saved with traffic aggregation over less number of paths, consider the network illustrated in Procedure 1. The paths $P_{1,2}^{s,d}$ and $P_{1}^{s,d}$ are neighbors because they share the links $E_6$ and $E_9$. Assuming that these paths have large enough capacity to carry their pair loads, the controller aggregates whole network traffic to $P_{1,2}^{s,d}$ and $P_{1}^{s,d}$. Then, it deactivates the remaining four core switches and six links to save energy.

### Procedure 1: Energy-aware path management algorithm

```
Procedure 1: Energy-aware path management algorithm

Input: Pair load $PUP$, pair capacity $\varphi(P^{s,d})$, pair energy $\varepsilon(P^{s,d})$

for $i = 1$ to $n$ do
  $V[i, \varphi(P^{s,d})] \leftarrow \varepsilon(P^{s,d})$
end for

for $i = 1$ to $n$ do
  for $c = \varphi(P^{s,d})$ to $\theta(P^{s,d})$ do
    if $c \cdot \varepsilon_i \geq \theta(P^{s,d})$ & $(V[i-1, c \cdot \varepsilon_i] - e_i) \leq (V[i-1, c])$ then
      $V[i, c] \leftarrow V[i-1, c \cdot \varepsilon_i] - e_i$
    else
      $V[i, c] \leftarrow V[i-1, c]$
    end if
  end for
end for
```

The first step of energy-aware path management is determination of the most energy efficient subset of paths of a pair $P^{s,d}$ while total capacity of $P^{s,d}$ ($\varphi(P^{s,d})$) is greater than or equal to its load ($\theta(P^{s,d})$). Time complexity of the brute force solution [generation of all combinations ($\theta(2^n)$)] and searching for the best combination ($\theta(n)$) are exponential. In this regard, we propose a polynomial time algorithm called energy-aware path management algorithm (EPMA) 1.

Let $\varphi(P^{s,d})$ denotes the pair capacity, $\theta(P^{s,d})$ denotes the pair load and $V[i, c] (1 \leq i \leq n$ and $\theta(P^{s,d}) \leq c \leq \varphi(P^{s,d}))$ denote the minimum energy consumption for a subset of paths for number of $i$ path and $c$ capacity. Assuming that all the paths are initially active, EPMA decides if the $i$-th path should be active or passive, at each step. In the former case, $V[i, c]$ is equal to $V[i-1, c]$. This means energy consumption and capacity stays the same. In the latter case, $V[i, c]$ becomes $V[i-1, c - \varepsilon_i] - e_i$. More clearly, current capacity $c$ reduces by $\varepsilon_i$ and energy consumption reduces by $e_i$. We can find the optimal solution by filling the table $V[0..n, \varphi(P^{s,d})]$. In this sense, entry $V[n, \theta(P^{s,d})]$ becomes the optimal solution. According to our problem, $V[i, c] = \infty$ for $0 \leq c \leq \theta(P^{s,d})$ and $V[0, c]$ is equal to maximum energy consumption for $0 \leq c \leq \theta(P^{s,d})$.

$$PDV(P^{s,d}) = \begin{cases} 
2(PUR(P^{s,d}) - PP)^+ / \overline{m}, & \text{if } PUR(P^{s,d}) \leq PP \\
2(PUR(P^{s,d}) - PP)^+ / \overline{m}, & \text{if } PUR(P^{s,d}) > PP 
\end{cases}.$$

(1)

$\forall s$ and $d \in V, s \neq d, k \in P^{s,d}$
The controller computes the energy consumption value of a path $P_{k}^{s}$ by multiplication Passive Neighbour Ratio (PNR) and Pivot Distance Value (PDV). PNR and PDV of a path $P_{k}^{s}$ is indicated as PNR($P_{k}^{s}$) and PDV($P_{k}^{s}$), respectively. PNR ($P_{k}^{s}$) is the ratio of passive neighbour paths over all of its paths. PDV($P_{k}^{s}$), however, depends on both utilization ratio ($\mu(P_{k}^{s}) = \vartheta(P_{k}^{s}) \times 100\varphi(P_{k}^{s})$) and pivot point (PP) as defined in Equation 1. PP is a predefined utilization ratio (e.g. 70%). It allows the controller to aggregate the desired amount of load to the path as much as possible. For instance, defining 70% forces the controller to reach the desired utilization as fast as possible.

Time complexity of EPMA is $O(n^*|\varphi(P_{k}^{s}) - \vartheta(P_{k}^{s})|)$. Notice that we design EPMA for a single pair. Therefore, total time complexity for the whole network becomes $O(n^*|\varphi(P_{k}^{s}) - \vartheta(P_{k}^{s})| \times B)$ where $B$ is number of pairs. As soon as the controller determines future active and passive paths, it may need to shift flows from the paths that will be passive to the paths that will remain active or just have been activated. In such cases, the controller determines current and new states of the paths first. To be clear, there are four states which are defined in Table 3. If there are paths in State 1, the controller shifts the flows in these paths to the paths that are in State 0 or State 2.

After the shifting process, the controller tries to deactivate all the network devices along the paths that do not share switches and links along their way with other paths. Since the whole path is passive, energy-saving increases dramatically. If this is not the case, the controller tries to deactivate the switches and links along its way that does not carry traffic. Before deactivating a switch, the controller makes sure that all the links coming and leaving the switch does not carry traffic. In addition to this, two new configuration values can be added to the configuration field of OFPT_PORT_MOD message for activating or deactivating ports.

EPMA finds the optimum path combination per pair assuming that all the paths are active. However, some of the paths may be in passive mode in reality and their capacity may be taken by neighbour paths. To handle this problem, ESM computes future capacities of passive paths because a passive path can be selected as a result of EPMA. To do so, it calculates the unused capacity of each link by subtracting load from its capacity and shares this unused capacity equally among the passive paths that the link serves. The capacity of the passive path becomes the capacity of the link which has the minimum capacity compared to other links that serve the same passive path. Let us show this procedure in an example for clarification purposes. Suppose that ESM executes EPMA for the pair ISS1-ESS1 as illustrated in Figure 1. Also, suppose that capacities of all links are 1Gbps and paths except $P_{1}^{2}$ and $P_{2}^{1}$ are active. The controller must find the future capacity of Path2 in case EPMA selects it. There are one, two and two passive paths on links $E_{2}$, $E_{6}$ and $E_{9}$, respectively. Load of $E_{2}$, $E_{6}$ and $E_{9}$ are all 0 because $P_{1}^{2}$ and $P_{2}^{1}$ do not carry traffic. Unused capacities of $E_{2}$, $E_{6}$ and $E_{9}$ are 1 Gbps for each. In this case, capacity share of $P_{2}^{1}$ for the links $E_{2}$, $E_{6}$ and $E_{9}$ are 1, 0.5 and 0.5 Gbps, respectively. The minimum of these capacities is 0.5 Gbps. Therefore, the future capacity of $P_{2}^{1}$ becomes 0.5 Gbps.

### 4.2 Adaptive load balancing

**Procedure 2:** Active balancing algorithm

```plaintext
pairCost = 0
for Path $P_{k}$ in $P^{d}$ do
    if isActive($P_{k}$) then
        pairCost = pairCost + $\lambda(P_{k})$
    end if
end for
for Path $P_{k}$ in $P^{d}$ do
    if isActive($P_{k}$) then
        if $\lambda(P_{k}) < \mu(V_{s}, V_{d})$ then
            addPathToList(highlyUtilizedPathList, $P_{k}$)
        else if $\lambda(P_{k}) > \mu(V_{s}, V_{d})$ then
            addPathToList(lightlyUtilizedPathList, $P_{k}$)
        end if
    end if
end for
for Path $P_{k}$ in highlyUtilizedPathList do
    flowPool = getFlows($P_{k}$)
    sortFlows(flowPool)
    for Path $P_{k}$ in lightlyUtilizedPathList do
        for Flow $f_{k}$ in flowPool do
            if shifting flow $f_{k}$ to $P_{k}$ makes $\lambda(P_{k}) < \mu(V_{s}, V_{d})$ then
                shiftFlow($f_{k}$, $P_{k}$, $P_{k}$)
            end if
            if $\lambda(P_{k}) > \mu(V_{s}, V_{d})$ then
                addPathToList(pathRemoveList, $P_{k}$)
                continue;
            end if
        end for
    end for
end for
for Path $P_{k}$ in pathRemoveList do
    removePath(pathRemoveList, $P_{k}$)
end for
for Path $P_{k}$ in pathRemoveList do
    if $\lambda(P_{k}) < \mu(V_{s}, V_{d})$ then
        continue;
    end if
end for
```

**Table 3** All state changes of paths after EPMA

| States  | Current state | Next state |
|---------|---------------|------------|
| State 0 | Active        | Active     |
| State 1 | Active        | Passive    |
| State 2 | Passive       | Active     |
| State 3 | Passive       | Passive    |

Abbreviation: EPMA, energy aware path management algorithm.
The controller performs load balancing adaptive to the traffic for two reasons. First of all, the controller has limited computational resources. Thus, computationally intensive actions may degrade the performance of the controller such as optimal (network-wide) load balancing. Second, optimum load balancing may not last a time because traffic is highly dynamic. Due to the aforementioned reasons, the controller balances the load of a pair among its active paths by equalising their costs (i.e. load, utilization, congestion level). Although paths are virtual and individual, pair-based load balancing converges to network-wide load balancing. This is because cost computation is based on the cost of physical links.

Active balancing algorithm is presented in Algorithm 2. In the first step of load balancing, the controller calculates the cost of active paths and pair equilibrium cost ($\mu(V_s, V_d)$). The pair equilibrium cost is the cost of all the paths converges to it after the load balancing process. As a path consists of links, we define path cost as the summation of costs of links which form the corresponding path. We denote the cost of a path $P_{k}^{d}$ and cost of one of its link $E_{k}^{d,k}$ as $\lambda(P_{k}^{d})$ and $\lambda(E_{k}^{d,k})$, respectively. In the second step, the states of paths are identified based on $\mu(V_s, V_d)$. A path whose cost is greater than $\mu(V_s, V_d)$ is classified as overloaded. Similarly, a path whose value is less than $\mu(V_s, V_d)$ is classified as under-loaded. If a path has the same cost as $\mu(V_s, V_d)$, it is already balanced. In the third step, the algorithm obtains information such as ISS, ESS, and PID regarding each flow in unbalanced paths. Finally, some of these flows are shifted from overloaded paths to under-loaded paths. The controller selects which flow to be shifted in a greedy manner (e.g. heavily to lightly loaded flow). Flow shifting occurs via updating the PID of flows defined in forwarding rules in edge switches.

The computation of pair equilibrium cost takes $O(n)$. Sorting flows by size takes $O(f^{*}\log f)$ where $f$ is number of flows. Shifting flows to underutilised paths takes $O(n^{*}\log f)$. Thus, time complexity of the active load balancing algorithm is $O(n^{*}(f^{*}\log f + n^{*}f))$ in the worst case where all paths are active. The controller performs load balancing for all pairs. Therefore, time complexity for overall network load balancing becomes $O(B^{*}n^{*}(f^{*}\log f + n^{*}f))$.

### 4.3 Active capacity resizing

Traffic load of some active paths may be higher than other active paths even when load balancing exists. This is because pair based load balancing takes some time to balance overall network load. As paths are actually virtual, and their capacities are re-sizeable. Dynamically adjusting active path capacities prevents the network from congestion and increases resource utilization. Note that, this process is performed on PSDB, not in data plane. For this reason, the controller does not explicitly alter rule of paths on switches.

As aforementioned, neighbour paths may share physical links. A physical link is split to one or more virtual links where each of which serves to a different path of different pairs. We denote a virtual link that belongs to a path $P_{k}^{d}$ as $E_{k}^{d,k}$. We denote a path $P_{k}^{d}$ on a link $E_{i}$. Active capacity resizing algorithm computes new virtual link capacity $\phi(E_{i}^{d,k})$ for all active links and paths proportional to loads as defined in the equation below. Subsequently, the algorithm iterates over all active paths to compute new capacity that is determined by the virtual link with the minimum capacity.

$$\phi(E_{i}^{d,k}) = \phi(E_{i}) \times \frac{\theta(P_{i})}{\theta(E_{i})}$$

(2) $\forall s and d \in V, s \neq d, k \in P_{s}^{d}, i \in E$

Although the computation of new path capacity is simple, the controller iterates over all virtual links and paths twice. Thus, the time complexity of the resizing algorithm becomes $O((|E| * |P|)^2)$. This indicates that resizing operation is a heavy process. For this reason, PRM does not perform resizing frequently unless the overall network is heavily loaded (e.g. 80% of its total capacity). In a heavier loaded network (e.g. 95% of its total capacity), the controller performs consecutive resizing operations because the threshold is exceeded even after the resizing operation. This infers that active capacity resizing must be avoided in such cases. To handle this problem, the controller stops resizing if the number of consecutive resizing operations in a predefined time interval exceeds a threshold (e.g. 3). In this case, PRM backs off so that ESM takes over to expand the capacity, if possible. Therefore, PRM and ESM work in harmony to increase resource utilization.

**Procedure 3: Active capacity resizing algorithm**

```plaintext
for Link $E_{i}$ in E do
  if isActive($E_{i}$) then
    $\phi(E_{i}) = \text{getLinkCapacityFromLSDB}(E_{i})$
    $\theta(E_{i}) = \text{getCurrentLinkLoadFromLSDB}(E_{i})$
    for Path $P_{k}$ in getPathsFromPSDB($E_{i}$) do
      if isActive($P_{k}$) then
        $\theta(P_{k}) = \text{getCurrentPathLoadFromPSDB}(P_{k})$
        $\phi(E_{i}^{d,k}) = \phi(E_{i}) \times \frac{\theta(P_{k})}{\theta(E_{i})}$
      end if
    end for
  end if
end for
for all Path $P_{k}^{d}$ do
  if isActive($P_{k}^{d}$) then
    linksOfPathList = getLinksFromPSDB($P_{k}^{d}$)
    temporaryMinimumCapacity = maximum link capacity
    for all Link $E_{i}^{d,k}$ in linksOfPathList do
      if isActive($E_{i}^{d,k}$) then
        if temporaryMinimumCapacity > $\phi(E_{i}^{d,k})$ then
          temporaryMinimumCapacity = $\phi(E_{i}^{d,k})$
        end if
      end if
    end for
    $\phi(P_{k}^{d}) = \text{temporaryMinimumCapacity}$
  end if
end for
```
5 EXPERIMENTAL RESULTS

In this section, we present the experimental results of our model. The proposed model is implemented using Floodlight [29] as a controller, Mininet [30] as a network emulator, Open vSwitch [31] as an SDN capable switch, TG [32] as a traffic generator, and OpenFlow [33] as a signalling protocol. Several scenarios have been created and various tests have been carried out in order to evaluate the proposed model using different topologies and traffic tracks [8].

The test scenarios are performed to analyse the effect of path number and traffic load on energy-saving performance, the effect of node connectivity on energy-saving performance, and the comparison of the proposed model with existing work in terms of energy saving.

5.1 Effects of path number per pair and traffic load on energy saving performance

In this test case, we have investigated the energy-saving performance of the proposed model for a various number of paths per pair under light, moderate, and heavy traffic loads. Path number per pair is 2 (P2) and 5 (P5). Net-M is chosen as the test topology. There are three test runs. During each run, active link and switch ratios are recorded, and the average of each run is calculated. Figures 2 and 3 illustrate energy-saving performance of P2 and P5 in terms of link and switch.

We have observed that active link ratio drops down to around 50%–60% in P2 for all traffic loads as illustrated in Figure 2. This experiment results in lower power consumption in P5 for light and moderate traffic loads since the active link ratio drops down to around 35%–45% at the end of the experiment. Note that, the energy-saving performance of both P2 and P5 is the closest to each other under heavy traffic load because the controller needs more resources to carry the heavy load. Also, notice that the active link ratio of P2 decreases to around 60%–70% in almost 50 s compared to P5. This ratio does not change much since then. The reason is that the controller has less choice in terms of path number per pair. However, the energy-saving performance of P5 is better after 300 s. The difference between the active switch ratio of P2 and P5 is much more apparent as shown in Figure 3 for all traffic loads. P2 shows poorer performance even under light traffic load. In case of heavy traffic load, the controller fails to make any switch passive. However, this is not the case for P5. The active switch ratio of P5 drops down to around 60% for light and moderate traffic loads. Although its energy-saving performance under heavy traffic load is not as good as others, the controller successfully makes a few amount of the switches passive.

5.2 Effect of connectivity on energy saving performance

In this test case, we have investigated the energy-saving performance of the proposed model for different topologies. The average degree of connectivity for NSFNET, Net-M, and Net-L are 3, 6, and 3.53, respectively. Path number per pair is two, and a moderate traffic load is applied. There are three test runs. During each run, active link and switch ratios are recorded, and the average of each run is calculated.

We have observed that the degree of connectivity does not affect the energy-saving performance of the proposed model on links as illustrated in Figure 4. The active link ratio in all topologies is close to each other. On the other hand, this is not the case for switches. NSFNET has the best performance among other topologies because paths in this topology have more common links and switches. Net-L has the worst performance among others because paths in Net-L are more diverse than others due to the high connectivity. This test case shows that the establishment of paths affects the energy-saving performance of the proposed model.

5.3 Comparison of energy saving performance

In this test case, we have investigated the energy-saving performance of the proposed model by comparing it with the approach defined in Reference [16]. In our model, as aforementioned, we define path energy consumption as the multiplication of PNR and PDV. In Reference [16], authors define energy consumption based on link utilization and a congestion threshold, that is 80%. We call this model the CARE model. In this test case, we run EPMA both for the proposed method and CARE model. Path number per pair is 2 (P2) and 5 (P5). Net-M is chosen as the test topology. Moderate traffic is applied in all three test runs. During each run, active link and switch ratios are recorded and the average of each run is calculated. Figures 5 and 6 illustrate energy-saving performance of the proposed and CARE models in terms of link and switch.

On one hand, the two models have almost the same performance for P2 in terms of active link ratio case as depicted in Figure 5. On the other hand, the proposed model outperforms the CARE model in the P5 case. Similarly, the results of the active switch ratio test resemble active link ratio test. The two models have almost the same performance for P2, and the proposed model outperforms the CARE model for P5.

5.4 Comparison of load balancing performance

In this test case, we have investigated the load balancing performance of the proposed model by comparing it with the previous model SRRM [8]. In this model, load balancing is performed among active paths. Path number per pair is 2 (P2) and 5 (P5). Net-M is chosen as the test topology. Moderate traffic is applied in all three test runs. During each run, load rates are recorded and the average of each run is calculated. Figure 7 illustrates load balancing performance of both models.
In both P2 and P5, the utilization of links is higher in ERM compared to SRRM. This is because the same load is distributed over less capacity due to passive paths in ERM. Additionally, utilization increases in line with energy-saving. In previous test cases, we have observed that energy-saving performance increases conforming to path number.
per pair. Hence, load utilization of P2 7a is lower than P5 7b. The trade-off between energy saving and load balancing indicates that high energy saving may cause over-utilization. Thus, this may result in significant network congestion. Finally, this test case shows that an adequate path number per pair and PDV selection allows efficient usage of network resources in terms of both energy saving and resource utilization.
CONCLUSION AND FUTURE WORKS

To put in a nutshell, this study proposes an ERM for SDN-based SPNs. In the proposed model, multi-paths are pre-established between each EP. This turns a complex physical network into a simple virtual one. The controller performs energy-saving, load balancing, and capacity updates based on this network. To save energy, the controller aggregates flows that belong to an EP into less number of paths, while taking neighbour paths and load level into account. Pair based power saving converges to global power saving in time. Apart from energy saving, the controller also performs adaptive load balancing by equalising path costs of individual pairs and adjusts path capacities based on network congestion even among active paths for further resource utilization.

In future work, we aim to increase the energy-saving performance of the proposed model using deep learning techniques. Additionally, we also aim to increase load balancing performance by estimating incoming traffic.

ACKNOWLEDGEMENTS

This research was supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. 113E253.

ORCID

Mahmut Rasih Çelenlioğlu https://orcid.org/0000-0001-8061-419X

REFERENCES

1. Gerber, A., Doverspike, R.: Traffic types and growth in backbone networks. In: Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fibre Optic Engineers Conference, pp. 1–3 (2011)

2. Alomari, S.A., Sumari, P.: Statistical information of the increased demand for watch the VOD with the increased sophistication in the mobile devices, communications and internet penetration in Asia. Corr, abs/1112.2042. http://arxiv.org/abs/1112.2042 (2011)

3. Chabarek, J., et al.: Power awareness in network design and routing. In: IEEE INFOCOM 2008 - The 27th Conference on Computer Communications (2008)

4. A gravid, M.K., Neama, G., Rafique, Y.: The impact of practical network constraints on the performance of energy-aware routing schemes. In: IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), pp. 77–81 (2015)

5. Tuysuz, M.F., Ankarali, Z.K., Gözüpek, D.: A survey on energy efficiency in software defined networks. Comput. Netw. 113, 188–204 (2017)

6. Van Heddeghem, W., et al.: Trends in worldwide ICT electricity consumption from 2007 to 2012. Comput. Commun. 50, 64–76 (2014). https://doi.org/10.1016/j.comcom.2014.02.008

7. McKeown, N.: Software-defined networking. In: Proceedings of IEEE INFOCOM'09, INFOCOM Keynote Talk, Brazil (2009)

8. Çelenlioğlu, M.R., Tuysuz, M.F., Mantar, H.A.: An SDN-based scalable routing and resource management model for provider service networks. Int. J. Commun. Syst. 31(8), e3530 (2018). https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.3530

9. Çelenlioğlu, M.R., Goger, S.B., Mantar, H.A.: An SDN-based energy-aware routing model for intra-domain networks. In: 2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 61–66 (2014)

10. McKeown, N., et al.: Openflow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38(2), 69–74 (2008). http://doi.acm.org/10.1145/1355734.1355746

11. Tucker, R., et al.: Energy consumption in IP networks. In: 2008 34th European Conference on Optical Communication (ECOC), pp. 1–8 (2008)

12. Heller, B., et al.: ElasticTree: saving energy in data centre networks. In: Proceedings of the 7th USENIX Conference on Networked Systems design and implementation. NSDI'10. Berkeley, CA, USA: USENIX Association, pp. 17–17. http://dl.acm.org/citation.cfm?id=1855711.1855728 (2010)

13. Gupta, M., Singh, S.: Greening of the internet. In: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and protocols for computer Communications. SIGCOMM `03. New York, NY, USA: ACM, pp. 19–26 (2003). http://doi.acm.org/10.1145/863955.863959

14. Zemmouri, S., Vakilinia, S., Cheriet, M.: Let’s adapt to network change: towards energy saving with rate adaptation in SDN. In: 2016 12th International Conference on Network and Service Management (CNSM), pp. 272–276 (2016)

15. Wang, R., et al.: Energy-aware routing algorithms in software-defined networks. In: 2014 IEEE 15th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1–6 (2014)

16. Thaenchaikun, C., et al.: Mitigate the load sharing of segment routing for SDN green traffic engineering. In: International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), pp. 1–6 (2016)

17. Razmjoureh, R., Bakhshi, B.: Green traffic engineering in SDN. In: 2016 24th Iranian Conference on Electrical Engineering (ICEE), pp. 693–698 (2016)

18. Fernández-Fernández, A., Cervelló-Pastor, C., Ochoa-Adly, L.: A multi-objective routing strategy for QoS and energy awareness in software-defined networks. IEEE Commun. Lett. 21(11), 2416–2419 (2017)

19. Bolla, R., et al.: The green abstraction layer: a standard power-management interface for next-generation network devices. IEEE Internet Computing, 17(2), 82–86 (2013)

20. Bruschi, R., et al.: Green extension of openflow. In: 26th International Teletraffic Congress (ITC), pp. 1–6 (2014)

21. Awad, M.K., et al.: A greedy power-aware routing algorithm for software-defined networks. In: IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), pp. 268–273 (2016)

22. Gomes, R.L., et al.: State-aware allocation of reliable virtual software defined networks based on bandwidth and energy. In: 13th IEEE Annual Consumer Communications Networking Conference (CCNC), pp. 411–416 (2016)

23. Hu, Y., et al.: GreSDN: Towards a green software defined network. In: 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 1–6 (2016)

24. Rahnamay, M., et al.: A traffic and resource-aware energy-saving mechanism in software defined networks. In: 2016 International Conference on Computing, Networking and Communications (ICNC), pp. 1–5 (2016)

25. Eppstein, D.: Finding the k shortest paths. SIAM J. Comput. 28(2), 652-673 (1999). https://doi.org/10.1137/S0097539795290477

26. Luo, M., et al.: An adaptive multi-path computation framework for centrally controlled networks. Comput. Netw. 83, 30–44. http://www.sciencedirect.com/science/article/pii/S0169755215000432 (2015)

27. Abe, J.O., Mantar, H.A., Yayimli, A.G.: k-maximally disjoint path routing algorithms for SDN. In: Proceedings of the 2015 International Conference on Cyber-enabled Distributed Computing and knowledge Discovery (CyberC), pp. 499–508 (2015)

28. Chen, J., et al.: When software defined networks meet fault tolerance: a survey. In: International Conference on Algorithms and Architectures for Parallel Processing Cham: Springer International Publishing, pp. 351–368 (2015). https://doi.org/10.1007/978-3-319-27137-8_27
29. Izard, R.: Floodlight v0.91 Release Notes. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343490/Floodlight+v0.91, (2014)

30. Lantz, B., Heller, B., McKeown, N.: A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Hotnets-IX. New York, NY, USA: ACM, pp. 19:1–19:6 (2010)

31. Pfaff, B., et al.: The design and implementation of open vswitch. In: Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation. NSDI’15. Berkeley, CA, USA: USENIX Association, pp. 117–130. http://dl.acm.org/citation.cfm?id=2789770.2789779 (2015)

32. McKenney, P.E., Lee, D.Y., Denny, B.A.: Traffic Generator Software Release Notes, SRI International and USC/ISI Postel Center for Experimental Networking. (2002), [Online: http://www.postel.org/tg/tg2002.pdf]

33. Openflow Switch Specifications Version 1.0.0. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf. Open Networking Foundation, (2009)

How to cite this article: Çelenlioğlu MR, Mantar HA. Energy aware adaptive resource management model for software-defined networking-based service provider networks. IET Netw. 2021;10:88–100. https://doi.org/10.1049/ntw2.12006