Deep Learning-Based Traffic Prediction for Energy Efficiency Optimization in Software-Defined Networking

Xiangyi Chen, Xingwei Wang, Bo Yi, Qiang He, and Min Huang

***Abstract*—Extensive redundant links and dedicated network components are deployed in existing networks to avoid network congestion caused by peak traffic. These network components are underutilized, leading to an extreme waste of electrical energy and negative environmental consequences. Most energy efficiency schemes have little consideration of the subsequent traffic load as well as the impact of energy saving on network performance, which brings many challenges, such as frequent flow rerouting, continual device state transition, and network load imbalance. In this article, we propose an energy efficiency optimization framework based on traffic prediction in software-defined networking, aiming at reducing network energy consumption while ensuring communication quality. First, we design a real-time traffic prediction mechanism based on gated recurrent unit neural network of deep learning to capture the temporal characteristics of network traffic and provide data basis for the deployment of energy-saving strategies. Second, we propose a heuristic algorithm for energy efficiency optimization to balance flow demand and energy consumption and to achieve dynamic load balancing and energy saving. Finally, simulation is carried out through Ryu controller, Mininet, and TensorFlow. Experimental results show that our algorithm achieves about 47.71% reduction in terms of the overall energy consumption with good network load balancing.**

***Index Terms*—Deep learning, energy efficiency, load balance, software-defined networking (SDN), traffic prediction.**

I. Introduction

NETWORK traffic is growing explosively, which is related to the increasingly plentiful multimedia applications [1] and the development of emerging technologies, such as the Internet of Things [2], mobile edge computing [3], big data [4], etc. The rapid growth of network traffic has resulted in excessive equipment deployment, high energy consumption, and negative environmental consequences. To avoid network congestion caused by peak traffic and accommodate to burst traffic, many redundant links and dedicated devices are deployed in existing networks. These excessive devices are powered ON at all times, causing an extreme waste of electrical energy [5]. According to the latest studies, in 2018, global data centers consumed up to 205 TWh of electricity, and due to traffic demand, the number of hyperscale data centers is expected to increase from 338 in 2016 to 628 in 2021, with the growth rate of near doubling [6], [7]. The total energy consumption of information technology equipment increased accordingly to around 130 TWh in 2018. Although the energy efficiency of equipment has improved compared to the past, this is not enough to reverse the trend. Meanwhile, the greenhouse gas footprints of communications technology (CT) are growing according to the present investigation. By 2030, the CT industry could contribute up to around 23% of global carbon dioxide emissions for the worst-case scenario [8].

In addition, related research show that network components are often underutilized [9], [10], e.g., the average link utilization of the backbone network is only 30%–40%, while the utilization during off-peak hours may drop by three times or more [9]. Moreover, the energy consumption of devices under low-load has little significant difference with the full-load, which makes the energy consumption disproportionate to the traffic volume and wastes a lot of energy [10].

The energy costs have significantly increased the overall costs of networks, which has heightened the demand for exploring novel, green, and efficient network optimization technologies [11], [12]. Considerable research efforts have been devoted to energy efficiency optimization schemes in traditional networks, including re-engineering, dynamic adaptation, and sleeping/standby [13], [14]. However, due to the lack of centralized control, in traditional networks, it is difficult to implement device management and network protocol update, which is prone to load imbalance and severely affect quality of service. Software-defined networking (SDN) brings a new view to solve the problem of network energy efficiency, which implements logically centralized control and enables the network programmability [15], [16]. With a global network view, the controller can dynamically develop centralized forwarding strategies based on network status and turn off the idle devices for energy saving. The programmability enables the network to adapt to the changing environment by taking actions such as

Manuscript received May 22, 2020; revised June 18, 2020; accepted July 1, 2020. Date of publication July 31, 2020; date of current version December 9, 2021. This work was supported in part by the National Key R&D Program of China under Grant 2017YFB0801701, in part by the National Natural Science Foundation of China under Grant 61872073, and in part by the LiaoNing Revitalization Talents Program under Grant XLYC1902010. *(Corresponding author: Xingwei Wang.)*

Xiangyi Chen, Xingwei Wang, and Bo Yi are with the College of Computer Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: xiangyichen3746@gmail.com; wangxw@mail.neu.edu.cn; yibobooscar@gmail.com).

Qiang He is with the College of Medicine and Biological Information Engineering, Northeastern University, Shenyang 110819, China (e-mail: heqiangcai@gmail.cn).

Min Huang is with the College of Information Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: mhuang@mail.neu.edu.cn).

Digital Object Identifier 10.1109/JSYST.2020.3009315

repairing the failures of network components or modifying the network strategies.

The key challenge for an energy efficiency network is to deal with the tradeoff between energy-saving and network performance, and the uncertainty of future traffic requirements makes its implementation more difficult. The existing energy efficiency solutions in SDN are mainly divided into three subcategories, including traffic aware, ternary content addressable memory (TCAM) optimization, and network virtualization [12], [17]. Most previous studies of network energy efficiency are based on the knowledge of precise traffic requirements [5], [18]. However, in actual situation, such assumptions are invalid. One possible solution is to predict future traffic load based on historical traffic data, which provides data basis for energy saving and deploys network components in advance for traffic peaks or burst flow to ensure network performance. Since traditional traffic prediction methods are not good at capturing the nonstationary features of network traffic, the emerging deep learning technology has been used to predict the network traffic, and it shows great potential to extract the burstiness and long-range dependence of network traffic [19]–[21]. Moreover, most existing energy efficiency schemes have little consideration on the impact of energy saving on network performance, which causes many severe problems, such as frequent flow rerouting, continual device state transition, and network load imbalance.

To address the above-mentioned issues, in this article, we propose a traffic prediction-based energy efficiency optimization (TPEO) framework in SDN. The main idea of TPEO is to improve network energy efficiency according to traffic prediction, considering the tradeoff between network load and energy consumption. Specifically, the devices with low-load can be powered OFF to reduce energy costs. On the other hand, network devices can be powered ON in advance before the peaks or burst traffic to balance network load. Our contributions are threefold, which are summarized as follows.

1) We design an adaptive traffic monitoring mechanism, which can dynamically adjust the traffic monitoring period and obtain the SDN traffic load in real time. The cost of network monitoring is reduced while the accuracy of monitoring is ensured.
2) We devise a real-time SDN traffic prediction method based on deep learning. We construct a prediction model through offline training, then record and forecast network traffic in real time by the sliding window of time series.
3) We propose a novel online heuristic algorithm for SDN energy efficiency optimization based on traffic prediction, which coordinates link resources with changes of the network to achieve dynamic load balancing and energy saving. To the best of our knowledge, this is the first article in which the SDN network energy efficiency has been optimized online by using deep learning technology for traffic prediction.

The remainder of this article is organized as follows. In Section II, we review previous research and point out the novelties of our article. We present the network framework and energy consumption model in Section III. In Section IV, we introduce the process of training gated recurrent unit (GRU) neural network model and traffic prediction. The mathematical formulation and heuristic algorithm are described in detail in Section V. We evaluate the performance of the proposed framework through extensive simulation in Section VI and summarize the article in Section VII.

II. Related Work

The energy efficiency in SDN has been intensively investigated, and its key methods include traffic aware, TCAM optimization and network virtualization integration, etc. [17], [18], [22]–[29]. Tuysuz *et al.* [17] proposed that energy-centric SDN solutions brought numerous benefits, such as providing multiple ways to balance the traffic load and achieving fair-share routing solutions. In [22] and [23], a green abstraction layer-based SDN energy-aware method was proposed, which individually adjusted the energy state of the data plane components and deployed control policies according to the network device status. To explore the impact of SDN and network virtualization integration on energy consumption, Al-Kaseem *et al.* [24] studied the energy consumption problem based on a real testbed in heterogeneous networks. Experiments have proven that compared with traditional networks, the integration of software defined-network functions virtualization (SD-NFV) architecture provides dynamic and scalable deployment for applications and significantly reduces network energy consumption. Another method of SDN energy saving is to optimize the TCAM used in forwarding switches, which is an expensive and power-consuming rule space. In [25], Kao *et al.* proposed a dynamically updatable ternary segmented aging bloom filter to optimize TCAM. It consisted of a segmented aging bloom filters algorithm and a ternary prefix-tagging encoder and was proven to reduce the power of OpenFlow switches in SDN.

Some researchers focused on hybrid SDN network scenarios. In [26], Jia *et al.* proposed the scheme to optimize network energy consumption by partially deploying SDN switches under the budget constraint and designed the flow-level explicit path control scheme. It rerouted flow and closed unnecessary links and switches in the network to save energy. Wang *et al.* [18] also discussed the issue of energy saving in partially deployed SDN and proposed a heuristic algorithm based on the smallest closed sets and spanning trees. Misra et al. [27] proposed a situation-aware protocol switching scheme in software-defined wireless sensor networks. It met application-specific requirements of users and optimized network performance including energy consumption. Another similar work [28] was based on the management service interface (MSI) of the state machine in SDN-based wireless sensor networks, which introduced the active–sleep module state into the MSI and considered the node's ON/OFF states to reduce energy consumption. Aiming at the energy efficiency of data center networks with constraints of flow deadlines, Xu *et al.* [29] proposed a method to reduce energy consumption through flow scheduling and exclusive routing in SDN.

Although researchers from academia and industry have proposed many schemes to improve SDN energy efficiency, there are still some crucial issues. The existing energy-saving strategies in SDN are mainly based on the current network status,

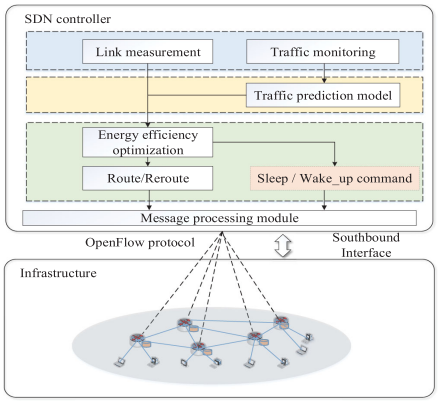


Fig. 1. Traffic prediction-based energy efficiency optimization framework in SDN.

TABLE I
COMMONLY USED NOTATIONS AND VARIABLES

Notation	Description
G,G'	Physical topology and virtual topology
V,V'	The set of all switches and the set of active switches
E,E'	The set of all links and the set of active links
Pv_j	Power cost of switch v_j
Pe_{base},Pe_{imax}	Base power and maximum power cost of link
$\omega(e_{ij})$,$C(e_{ij})$	Occupied link bandwidth and link capacity of e_{ij}
$U(e_{ij})$	Link utilization of e_{ij}
$S(v_j)$	Binary variable represents the switch v_j state
$W(e_{ij})$	Binary variable represents the link e_{ij} state
tx_bytes	Number of flow bytes transmitted by the port
f_{rate},$\Delta\gamma$	Flow rate and variation ratio of flow rate
γ_l,γ_h	Lower and upper bound of variation ratio of flow rate
T_{base}	Base monitoring period
$T_{\min}$,$T_{\max}$	Minimum and maximum monitoring period
TM^t	Links load matrix at time t
$\bar{U}(G')$	Average link utilization of virtual topology
$\delta(G')$	Standard deviation of link utilization in virtual topology
$E(G')$	Total power consumption of virtual topology
λ_u,λ_e	Load balancing factor and energy-saving factor
P_χ	Actual power consumption of the energy saving algorithm
P_O	Power consumption without energy saving
$\Gamma(G')$,$Cost(G')$	Network energy efficiency and cost

neither considering the future network load nor actively protecting network performance while energy saving. We have implemented real-time SDN traffic load forecasting and deployed online energy-saving strategies to balance flow demand and energy consumption for ensuring network transmission performance.

III. SYSTEM MODEL

A. Network Framework

The energy efficiency optimization framework based on traffic prediction proposed in this article is illustrated in Fig. 1. As network topology changes dynamically with the deployment of energy-saving strategies based on traffic load, the physical topology of the network at time T is denoted as $G(V, E, T)$, where V is the set of switches and v_i represents switch i, $v_i \in V$. E is the set of physical links with parameters such as link bandwidth and delay, and $e_{ij} \in E$ represents that there is a link between switches v_i and v_j. $G'(V,' E,' T)$ represents the virtual topology of the network at time T, which only includes switches and links powered ON in the network. The notations and variables used in our formulations are shown in Table I.

The SDN switch is composed of flow tables, secure channels, and OpenFlow protocol. When a new flow arrives, the switch performs field matching by checking the flow tables. If a match is found, the corresponding forwarding operation will be performed, otherwise, the flow is discarded or the decision is left up to the controller. As shown in Fig. 1, through the traffic monitoring and link measurement modules, the SDN controller periodically collects and maintains statistical information, including network topology, link load, and switch status. The traffic prediction model predicts the network load in real time through the collected flow data and passes the results to the energy efficiency optimization module. Based on the current and the predicted traffic load, the energy efficiency optimization module deploys the real-time energy-saving strategies and executes the sleep/wake-up commands. The path of routing/rerouting is calculated on the optimized network topology. SDN switches perform traffic measurement and send the measurement results to the controller. Meanwhile, switches need to update the flow tables according to the messages from the controller and perform data forwarding.

B. Energy Consumption Model

Since the network devices need to forward the arriving service flow, the energy consumption changes dynamically with the traffic load. We define binary variable $W(e_{ij})$ to indicate the status of e_{ij}, which is represented as follows:

$$W(e_{ij}) = \begin{cases} 1, \text{ if link } e_{ij} \text{ is in active} \\ 0, \text{ otherwise.} \end{cases} \tag{1}$$

Similarly, binary variable $S(v_j)$ indicates the status of v_j, which is defined in the following equation:

$$S(v_j) = \begin{cases} 1, \text{ if switch } v_j \text{ is in active} \\ 0, \text{ otherwise.} \end{cases} \tag{2}$$

According to [30], the switch mainly consists of a chassis, a number of line-cards, and ports. The power consumption incurred by the chassis and line-cards is basically fixed and independent of the traffic load, while the port power consumption is proportional to the utilization. We use the power consumption of the switch chassis and line-cards to represent the switch power consumption, which is expressed by Pv_j. Similarly, we use the port power to represent the link power, i.e., for each link, the link power is proportional to the link utilization. The power consumption of link e_{ij} can be expressed by

$$Pe_{ij} = Pe_{\text{base}} + (Pe_{\text{imax}} - Pe_{\text{base}}) * U(e_{ij}) \tag{3}$$

where Pe_{base} is the basic link power in active, Pe_{imax} is the maximum link power, and $U(e_{ij})$ is the link utilization. $U(e_{ij})$ is calculated as follows:

$$U(e_{ij}) = \frac{\omega(e_{ij})}{C(e_{ij})} \tag{4}$$

where $\omega(e_{ij})$ is the occupied link bandwidth of e_{ij} and $C(e_{ij})$ is the capacity of e_{ij}. The total network power consumption is the

sum of the switch power consumption and the link power consumption. The total network power consumption is expressed by

$$E = \sum_{v_j \in V} S(v_j) * Pv_j + \sum_{e_{ij} \in E} W(e_{ij}) * Pe_{ij}. \tag{5}$$

IV. Traffic Prediction

A. Traffic Monitoring

Traffic monitoring is the basic work in our framework, which is based on the granularity of data flow in SDN. Related studies have shown that OpenFlow provides convenient and precise traffic measurement without packet sampling or incurring any exorbitant overhead [31], [32]. It is shown in [33] that polling the last switch on the flow path has relatively high accuracy and low overhead. Therefore, we allocate the last switch on the flow path to be the point of reference to obtain the relatively accurate link load. The controller monitors the network flow by periodically polling flow bytes and packet counters from the last switch on the flow paths, while the underlying switches send the statistics information to the controller. Traffic monitoring records network traffic and stores it in the log file. On the other hand, the controller sends StatsRequest periodically through OpenFlow protocol to update the status of each port, which is the basic data of link measurement. Switches encapsulate the ev.msg containing state information in the reply message and send it to the controller.

The OpenFlow flow entry provides the number of transmitted bytes, which is described as tx_bytes. The flow rate is calculated by the difference between the numbers of bytes obtained from two measurements, divided by the time interval, which is expressed as follows:

$$f_{\text{rate}} = \frac{tx\_\text{bytes}_2 - tx\_\text{bytes}_1}{t_2 - t_1} \tag{6}$$

where t is the time stamp, which is accurate to milliseconds.

Although OpenFlow brings many benefits to flow monitoring during measurement, the controller still injects additional traffic in the form of probe packets by sending StatsRequest message periodically, which consumes the secure channel bandwidth from switches to the controller and causes monitoring overhead. Related study shows that frequent traffic monitoring is necessary only when traffic changes rapidly [33]. Consequently, we design a dynamic traffic monitoring mechanism to reduce the overhead of traffic measurement while ensuring the accuracy, which decreases the polling intervals as the stream rapidly changes and increases the polling interval when the flow statistics converge to a stable behavior. The flow monitoring period is calculated as follows:

$$\Delta\gamma = \frac{|f'_{\text{rate}} - f_{\text{rate}}|}{f_{\text{rate}} + \tau_1} \tag{7}$$

$$T_{\text{monitor}} = \begin{cases} \min\left\{\left\lfloor \dfrac{\lambda_1}{\Delta\gamma} \right\rfloor T_{\text{base}}, T_{\max}\right\}, & \Delta\gamma < \gamma_l \\ T_{\text{base}}, \gamma_l < \Delta\gamma < \gamma_h \\ \max\left\{T_{\min}, \left\lfloor \dfrac{\lambda_2}{\Delta\gamma} \right\rfloor T_{\text{base}}\right\}, & \Delta\gamma > \gamma_h \end{cases} \tag{8}$$

where f'_{rate} is the flow rate in current measurement, f_{rate} is the flow rate measured in the previous period, $\Delta\gamma$ is the rate of change, τ_1 is a nonzero positive number, γ_l and γ_h are the thresholds of the rate of change for adjusting the monitoring period, λ_1 and λ_2 are the period control factors, T_{base} is the monitoring reference period, $T_{\min}$ is the lower limit of the monitoring period, and $T_{\max}$ is the upper limit of the monitoring period. The monitoring period decreases while the rate of change increases. In addition, it is always bounded by the thresholds of the monitoring period. The dynamic monitoring period can reduce the flow measurement overhead while ensuring the accuracy of flow monitoring.

B. GRU Neural Network Model Training

The network traffic prediction model requires more powerful abilities, including the abilities to extract the long-range dependency and deal with the high-computational situation of nonlinear data. Since traditional traffic prediction methods are unable to cope with this issue, as a method of deep learning, GRU has such abilities and can capture the dependencies of different time scales adaptively, which shows the great potentiality of traffic prediction [21]. In addition, as a special variant of recurrent neural network (RNN), GRU improves the internal unit and alleviates the problems such as gradient disappearance or explosion of traditional RNN [34]. Furthermore, compared with another variant long shot-term memory (LSTM), GRU simplifies the gate structure and neuron states and reduces model parameters, greatly improving training efficiency. Therefore, we adopt GRU neural network model for traffic prediction.

The input sequence of GRU is expressed as $X = (x_1, x_2, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_n)$, and the output sequence is $H = (h_1, h_2, \ldots, h_{t-1}, h_t, h_{t+1}, \ldots, h_n)$. W, U, and b indicate the input layer weight, hidden layer weight, and the bias vector, respectively. z_t, r_t, $\tilde{h}_t$, and h_t represent the state of update gate, the state of reset gate, activation function of update, and activation function of hidden state at time interval t, respectively. The GRU calculation method is as shown by the following:

$$z_t = \sigma\left(W_z x_t + U_z h_{t-1} + b_z\right) \tag{9}$$

$$r_t = \sigma\left(W_r x_t + U_r h_{t-1} + b_r\right) \tag{10}$$

$$\tilde{h}_t = \tanh\left(W_h x_t + U_h h_{t-1} r_t + b_h\right) \tag{11}$$

$$h_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t. \tag{12}$$

Training a predictive model includes the following steps.

1) Read the traffic data and construct the format of input data according to the requirements of GRU. In this article, the traffic matrix is transposed to Vector_TM^t.

2) Normalize the data and divide the historical data into training set and test set. Here, the data of the latest windows are used as test set, and the remaining data are used as training set.

3) Build the prediction model and adjust model parameters during training. According to the above introduction of GRU model, the set of GRU training parameters (weights and biases)

can be represented by

$$\chi = (W_r, W_z, W_h, U_r, U_z, U_h, b_r, b_z, b_h). \tag{13}$$

4) In back-propagation training, the validity of the training model can be measured by calculating the error between the prediction $\tilde{f}_{ij}^t$ and the real data f_{ij}^t. We use mean square error (MSE) as the cost for back-propagation, which is expressed as

$$\text{Cost} = \text{MSE} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\tilde{f}_{ij}^t - {f^t}_{ij}\right)^2. \tag{14}$$

The training goal is to learn parameters dynamically from historical data and minimize the value of MSE for obtaining the optimal parameter set $\mathbf{X}$. The process of optimizing parameters can be expressed as follows:

$$\mathbf{X} = \arg\min_{\chi} \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left(\tilde{f}_{ij}^t - {f^t}_{ij}\right)^2. \tag{15}$$

5) After Cost is obtained, the model performs back-propagation through stochastic gradient descent to update the parameter set χ in the network. We apply the adaptive moment estimation (Adam) optimizer [35] to cost optimization. As a method to calculate the adaptive learning rate of parameters, Adam combines the advantages of Momentum and RMSProp to optimize parameters and minimize the cost of each layer. The process is shown in (16)–(21). The process of gradient computing in the objective function $f(\chi)$ for parameter χ is as shown in the following equation:

$$\nabla\text{Cost}_t = \nabla_\chi f_t(\chi_{t-1}). \tag{16}$$

In this article, our training goal is to minimize the MSE. ∇Cost_t is the gradient vector of cost function which is the partial derivative of $f_t(\chi)$ for parameter χ estimated at time step t.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) * \nabla\text{Cost}_t \tag{17}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) * \nabla\text{Cost}_t^2 \tag{18}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{19}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{20}$$

$$\mathbf{X}_t = \mathbf{X}_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t \tag{21}$$

where β_1 and β_2 are exponential decay rates, and $\beta_1, \beta_2 \in [0, 1)$. ϵ is a constant. m_t and v_t are the first moment vector and the second moment vector, respectively. $\hat{m}_t$ and $\hat{v}_t$ are the bias correction of m_t and v_t, and η is the learning rate. Finally, the parameters are updated by (21).

C. Traffic Prediction Mechanism

In the network with n nodes, the link load at time t is represented by n-order matrices TM^t. Specifically, ${f_{ij}}^t$ represents the traffic from switch i to switch j at time t, and the network link load matrix is expressed as follows:

$$TM^t = \begin{bmatrix} {f_{11}}^t & \cdots & {f_{1j}}^t & \cdots & {f_{1n}}^t \\ \vdots & & \vdots & & \vdots \\ {f_{i1}}^t & \cdots & {f_{ij}}^t & \cdots & {f_{\text{in}}}^t \\ \vdots & & \vdots & & \vdots \\ {f_{n1}}^t & \cdots & {f_{nj}}^t & \cdots & {f_{nn}}^t \end{bmatrix}. \tag{22}$$

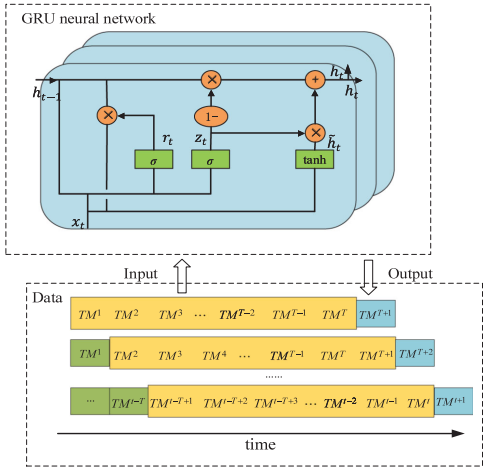


Fig. 2. GRU traffic prediction model based on time sliding window.

We use the time sliding window approach to perform traffic prediction, which uses the historical traffic data with the window as input and the prediction result as output. Moreover, with T time steps, the number of the collected n-order matrices TM^t is T, which are denoted as TM^T, and $TM^T = (TM^t, TM^{t-1}, TM^{t-2}, \ldots, TM^{t-T+2}, TM^{t-T+1})$. Here, the traffic prediction model is expressed as predicting TM^{t+1} by $(TM^t, TM^{t-1}, TM^{t-2}, \ldots, TM^{t-T+2}, TM^{t-T+1})$, where T is the prediction window length.

In order to make better use of deep learning models, we convert the traffic matrix TM^t into vector, Vector_$TM^t = (f_{11}^t, f_{12}^t, f_{13}^t, \ldots, f_{ij}^t, \ldots, f_{n(n-1)}^t, f_{nn}^t)$, which denotes the traffic vector at time t. f_{ij}^t in TM^t corresponds to the kth item in Vector_TM^t, $k = (i-1)\times n + j$. Therefore, the GRU traffic prediction model is transformed to predict Vector_TM^{t+1} by (Vector_TM^t, Vector_TM^{t-1}, Vector_TM^{t-2}, $\ldots$, Vector_TM^{t-T+2}, Vector_TM^{t-T+1}).

As shown in Fig. 2, the main idea of GRU traffic prediction model is to extract the timing characteristics of network traffic data by training GRU neural network model and to predict the network traffic of the next timing window in real time using the traffic matrix data constructed through traffic monitoring. Meanwhile, the sliding windows gradually move forward over time. According to the traffic data predicted by the GRU prediction model, the total network traffic and local traffic can be estimated, which provides an effective data basis for optimizing the energy efficiency of the network.

V. Energy Efficiency Optimization

A. Objective Function

According to the statistical messages between the controller and switches through OpenFlow protocol, the average link utilization of the virtual topology is described as follows:

$$\bar{U}(G') = \frac{\sum_{e_{ij} \in E'} U(e_{ij})}{|E'|}. \tag{23}$$

The standard deviation of link utilization is an important indicator, which characterizes the load balance degree of the network. In the TPEO framework of this article, the standard deviation of link utilization in virtual topology is represented as

$$\delta(G') = \sqrt{\frac{\sum_{e_{ij} \in E'} \left(U(e_{ij}) - \overline{U}(G')\right)^2}{|E'|}}. \tag{24}$$

In this article, our goal is to jointly optimize network energy consumption and load balancing to improve network energy efficiency $\Gamma(G')$. To be specific, the standard deviation of network link utilization indicates the degree of network load balancing, which is defined in (24), while energy consumption shows the power cost of the entire network and is defined in (5). We normalize the standard deviation $\delta(G')$ and the total power consumption $E(G')$ to $\hat{\delta}(G')$ and $\hat{E}(G')$, respectively. The cost function of the virtual topology is expressed as $\text{Cost}(G') = \lambda_u \hat{\delta}(G') + \lambda_e \hat{E}(G')$. Correspondingly, the energy efficiency of the virtual topology is defined as

$$\Gamma(G') = \frac{1}{\lambda_u \hat{\delta}(G') + \lambda_e \hat{E}(G')} = \frac{1}{\text{Cost}(G')} \tag{25}$$

where λ_u is the load balancing factor and λ_e is the energy-saving factor, $\lambda_u + \lambda_e = 1$, $\lambda_u, \lambda_e \in [0, 1]$. Our objective is to maximize the energy efficiency of the virtual topology, i.e., to minimize the $\text{Cost}(G')$, which is expressed as

$$\min \text{Cost}(G') \tag{26}$$

subject to

$$\omega(f_{ij}) + \omega(e_{ij}) \leq C(e_{ij}) \tag{27}$$

$$\sum f(e_{ij}) \leq W(e_{ij}) * C(e_{ij}) \tag{28}$$

$$f_{ij} \geq 0 \tag{29}$$

$$S(v_j), W(e_{ij}) \in \{0, 1\} \tag{30}$$

$$U(e_{ij}) \leq U_{\max} \tag{31}$$

$$W(e_{ij}) \leq S(v_i) \tag{32}$$

$$W(e_{ij}) \leq S(v_j) \tag{33}$$

$$\lambda_u, \lambda_e \in [0, 1]. \tag{34}$$

Here, (27) shows the bandwidth capacity constraint, where f_{ij} is the new flow in link e_{ij} and $\omega(e_{ij})$ is the occupied bandwidth. Equation (28) guarantees that the traffic in the dormant links is 0, and the sum of $f(e_{ij})$ is the total traffic in link e_{ij}. In (29), traffic constraint is described, i.e., each flow in the link is greater than or equal to 0. Equation (30) indicates that the status of switches and links are ON or OFF. Equation (31) is the load balancing constraint. Equations (32) and (33) indicate when the switch is sleep and the links connected are sleep. Equation (34) is a weight coefficient constraint. In fact, load balancing and energy saving are mutually exclusive. Sleeping switches and links can reduce network energy consumption, but, correspondingly, the load of active links will increase, which may cause uneven load and affect network performance. On the other hand, turning more links and switches ON can alleviate the load of active links and ensure the effective transmission of traffic, and also at the cost of increasing energy consumption. The time to find the optimal solution in this problem is intolerable, especially in large-scale networks. The heuristic algorithm can find the approximate optimal solution within a limited time, which is crucial for our real-time traffic prediction and energy efficiency optimization. Therefore, we propose a heuristic algorithm to solve the problem.

B. Heuristic Algorithm

Network energy efficiency optimization is responsible for turning ON or turning OFF the corresponding switches and links to reduce network energy consumption while meeting traffic demands. As introduced in Section III, the framework needs to maintain two topologies. One is the physical topology with all switches and links, and the other is the virtual topology, which only contains the links and switches turned ON in the network.

In the proposed energy efficiency optimization, we consider the regularity of network traffic, and our goal is to coordinate network link resources and network energy consumption with the dynamic changes of the network. Therefore, we propose a heuristic algorithm that maintains and updates the virtual topology $G'(V,' E')$ based on the predicted traffic matrix TM^{t+1}. The main idea of our algorithm is as follows: When it is predicted that the network traffic is small or there are idle links in the network for the next period, the sleep strategy is triggered and the sleep command is executed according to the sleep priority. Meanwhile, when the network traffic is predicted to be heavy or the local link is predicted to be overload for the next period, the wake-up strategy is triggered and the wake-up command is executed according to the wake-up priority. In this situation, switches and links are deployed in advance to deal with traffic peaks or burst flow. Energy efficiency optimization is a topology optimization problem, which aims to find the smallest subset of the network to meet the traffic demand over the next period. When the link load is more balanced and the network energy consumption is lower, the performance of the network is better.

Specifically, low-load and high-load links are added to the set l_f and the set h_f, respectively. Moreover, we use the average link utilization of the virtual topology to estimate the total network traffic, which is represented by $\bar{U}(G')$. Then, the algorithm deploys the sleep/wake-up strategies based on current network traffic and predicted traffic conditions and makes a smooth transition through weak wake-up or weak sleep modes. Finally, after performing the energy efficiency optimization, the network state needs to be passed to the routing module for forwarding traffic data under the new topology. When routing, we first

Algorithm 1: Sleep Algorithm With Priority.

Input: $G(V,E)$, ${G'}_t(V'_t,E'_t)$, TM^{t+1}, T_k
Output: ${G'}_{t+1}(V'_{t+1},E'_{t+1})$

Initialize $T_k \leftarrow T_0$;
while $T_k > T_{k\min}$ **do**
  for $e_{ij} \in l_f$ **do**
    if $f(e_{ij})_t < f_l$ and $f(e_{ij})_{t+1} < f_l$, $e_{ij} \in {G'}_t(V'_t,E'_t)$ **then**
      delete(e_{ij}, E'_{t+1});
      if $\Gamma({G'}_{t+1}) > \Gamma({G'}_t)$ **then**
        SleepSet$_{\text{links}} \leftarrow e_{ij}$;
      else
        add$(e_{ij}, \text{SleepSet}_{\text{links}}, P)$,
        $P = \exp(-(\Gamma({G'}_t) - \Gamma({G'}_{t+1}))/\mu T_k)$;
      end if
    end if
    ${G_k} = {G'}_{t+1}$;
  end for
  $T_k = \delta T_k$;
end while
if $\Delta\bar{U}(G') > 0$ **then**
  $\eta_s = \lambda_{\text{low}}$;
  sleep_links(SleepSet$_{\text{links}}, \alpha_s(e_{ij})$);
else
  $\eta_s = \lambda_{\text{high}}$;
  sleep_links(SleepSet$_{\text{links}}$);
  sleep_nodes(SleepSet$_{\text{nodes}}, \alpha_s(v_j)$);
end if
for $v_j \in V'_{t+1}$ and NumPort$_j < 2$ **do**
  sleep_nodes(v_j);
end for
return Optimized topology ${G'}_{t+1}(V'_{t+1}, E'_{t+1})$

calculate the path on the virtual topology, which will not cause an increase in energy consumption. If the virtual topology cannot find a path that meets the conditions, the path is calculated on the global topology. Now, we need to increase energy consumption for flow demands.

1) Sleep Algorithm: The sleep algorithm proposed is based on the information of predicted traffic to perform topology optimization, and the optimization goal is to maximize network energy efficiency. Intuitively, the contribution of the sleep algorithm is to reduce network energy consumption. In this case, we design a flexible protection mechanism, i.e., only considering sleeping low-flow link e_{ij} when the flow of e_{ij} is less than f_l at t and $t+1$. Note that these low-flow links are not powered OFF directly. We need to judge whether it can enhance the energy efficiency of the network by sleeping strategy. If energy efficiency can be improved, we will add it to SleepSet$_{\text{links}}$. Otherwise, the operation is performed according to the probability $\exp(-(\Gamma({G'}_t) - \Gamma({G'}_{t+1}))/\mu T_k)$, which helps the algorithm to overstep the local optimum with a certain probability and approach the global optimum. T_k represents the optimization parameter, and μ is a constant for adjusting the probability. The algorithm halts iteration when T_k reaches $T_{k\min}$.

Specifically, the sleep strategy is divided into two modes: Sleep and weak sleep, where η_s is the sleep level factor. The deployment of the sleep strategy is related not only to the total volume of traffic but also the change rate of traffic. For example, although the network traffic is low, when it shows a upward trend, i.e., $\Delta\bar{U}(G') > 0$, we invoke weak sleep for protection, $\eta_s = \lambda_{\text{low}}$. Then, we focus on the links. We calculate sleep priority $\alpha_s(e_{ij})$ for links by (35), and only m links with higher priority are powered OFF. On the other hand, when the traffic is low and shows a downward trend, i.e., $\Delta\bar{U}(G') < 0$, the stronger sleep strategy is triggered, $\eta_s = \lambda_{\text{high}}$. At this time, we sleep all links of SleepSet$_{\text{links}}$ and sleep n nodes connected according to priority $\alpha_s(v_j)$, which reduces network energy consumption and further improves network energy efficiency

$$\alpha_s(e_{ij}) = \frac{\beta_1}{U(e_{ij})^{t+1} + \delta_1} + \frac{\beta_2}{\max\left\{\text{flow}(v_i^{t+1}), \text{flow}(v_j^{t+1})\right\} + \delta_2}. \tag{35}$$

The link sleep priority is related to the utilization of the link and the load of the connected switches. β_1 and β_2 are the sleeping link factor and the sleeping node factor, respectively. δ_1 and δ_2 are nonzero constants. When performing sleep strategy, we reset the forwarding weight of links in the set to 0, which means that they will not be selected for routing in the next period. In fact, the flow on the links needs to be forwarded or scheduled before the links being actually powered OFF. The node sleep priority is calculated as follows:

$$\alpha_s(v_j) = \frac{1}{\sum \text{flow}(v_j^{t+1}) + \delta_3} \tag{36}$$

where δ_3 is a nonzero constant. The node priority is inversely proportional to its load. Finally, we judge if the number of the open ports of a switch is less than 2, i.e., no flow reaches the switch. Under this circumstance, we power OFF the switch to reduce unnecessary power cost and maximize network energy efficiency. After executing the sleep algorithm, the topology of G' is optimized. The pseudocode of the sleep algorithm is shown in Algorithm 1.

2) Wake-up Algorithm: Similar to the sleep algorithm, the main purpose of the wake-up algorithm is to optimize the network energy efficiency based on the predicted traffic. Different from the sleep algorithm, the goal of the wake-up algorithm is to reduce the standard deviation of link utilization, which contributes significantly to load balancing.

The urgency of the wake-up algorithm is often higher than the sleep algorithm because it may degrade the network performance. Therefore, the wake-up strategy will be triggered when the flow of links is higher than f_h at time t or time $t+1$. e_{ij} belongs to the set of high flow link h_f. Different from the sleep algorithm which can directly locate the links that need to be powered OFF, the wake-up algorithm requires adding the connected link e_{ik} to the WakeupSet$_{\text{links}}$, i.e., other links of the same source switch, $e_{ik} \in G$, and $e_{ik} \notin G'$. Similarly, we randomly select a connected link and wake it up if it improves energy efficiency

Algorithm 2: Wake-Up Algorithm With Priority.

Input: $G(V,E)$, ${G'}_t(V'_t, E'_t)$, TM^{t+1}, T_k
Output: ${G'}_{t+1}(V'_{t+1}, E'_{t+1})$

Initialize $T_k \leftarrow T_0$;
while $T_k > T_{k\min}$ **do**
  for $e_{ij} \in h_f$ **do**
    if $f(e_{ij})_t > f_h$ or $f(e_{ij})_{t+1} > f_h$ **then**
      $e_{ik} \leftarrow$ random(e_{ij}), $e_{ik} \in$ LinkedSet$_{E'_t}$, and $e_{ik} \notin {E_t}'$;
      add(e_{ik}, E'_{t+1});
      if $\Gamma(G'_{t+1}) > \Gamma(G'_t)$ **then**
        WakeupSet$_{\text{links}} \leftarrow e_{ik}$;
      else
        add$(e_{ik}, \text{WakeupSet}_{\text{links}}, P)$,
        $P = \exp(-(\Gamma(G'_t) - \Gamma(G'_{t+1}))/\mu T_k)$;
      end if
    end if
    $G_k = G'_{t+1}$;
  end for
  $T_k = \delta T_k$;
end while
if $\Delta\bar{U}(G') > 0$ **then**
  $\eta_w = \beta_{\text{high}}$;
  wake_up(WakeupSet$_{\text{links}}$);
  wake_up(WakeupSet$_{\text{nodes}}, \alpha_w(v_k)$);
else
  $\eta_w = \beta_{\text{low}}$;
  wake_up(WakeupSet$_{\text{links}}, \alpha_w(e_{ik})$);
end if
for $e_{ik} \in E'$ and $v_i \notin V'_{t+1}$ or $v_k \notin V'_{t+1}$ **do**
  wake_up(v_i, v_k);
end for
return Optimized topology ${G'}_{t+1}(V'_{t+1}, E'_{t+1})$

of the network, otherwise, adding it to the wake-up queue with probability $\exp(-(\Gamma(G'_t) - \Gamma(G'_{t+1}))/\mu T_k)$.

Similarly, the wake-up strategy is divided into two modes: Wake-up and weak wake-up. η_w indicates the wake-up factor. The deployment of the wake-up strategy is also affected by the total network traffic and the rate of change. For example, when the network traffic is high and shows a downward trend, the wake-up algorithm will trigger the weak wake-up to ensure a smooth transition, $\eta_w = \beta_{\text{low}}$. We calculate link wake-up priority $\alpha_w(e_{ik})$ by (37), and wake-up m higher priority links. On the other hand, when the network flow is high and still represents an upward trend, it indicates the arrival of traffic peak. We implement wake-up strategy which is more forceful to cope with the increasing traffic demand, $\eta_w = \beta_{\text{high}}$. Now we power ON all links in the queue and n higher priority switches according to the wake-up priority $\alpha_w(v_k)$

$$\alpha_w(e_{ik}) = \sum \text{flow}({v_i}^{t+1}) + \sum \text{flow}({v_k}^{t+1}). \tag{37}$$

The link wake-up priority is associated with the load of the connected nodes. The greater load of the connected nodes means the higher priority. The calculation process of node wake-up priority is shown in the following equation:

$$\alpha_w(v_k) = \sum \text{flow}({v_i}^{t+1}). \tag{38}$$

Similarly, the node wake-up priority is proportional to the load of the connected node. Finally, we need to ensure that the node is powered ON since the connected link is powered ON to improve the robustness of the network. We obtain the optimized network topology after executing the wake-up algorithm. The pseudocode of the wake-up algorithm is shown in Algorithm 2.

TABLE II
SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
Pe_{base}	2W	λ_e	0.5
Pe_{imax}	4W	$T_{\min}$	1
Pv_j	310W	$T_{\max}$	20
$time\_step$	12	f_l	0.1
λ_u	0.5	f_h	0.8

VI. PERFORMANCE EVALUATION

To evaluate the performance of our proposed energy efficiency optimization algorithm based on traffic prediction, we conduct the experiments by the Mininet [36] network simulation platform, the extended Ryu [37] controller, and the Open vSwitch [38]. Moreover, traffic prediction is implemented based on TensorFlow [39] and Keras deep learning library [40]. We perform the experiments with reference to the Fattree topology in [41]. We first verify the feasibility of traffic prediction by the public data set of the Abilene backbone network [42]. The Abilene data set contains real network traffic data, which is sampled every 5 minutes for 6 months. Then, we encapsulate and customize the command of the Iperf in Mininet, which realizes the automatic packet-sent by reading the generated data stream file. The related parameters of simulation experiments are shown in Table II.

First, we compare GRU, LSTM, and RNN to evaluate the characteristics of deep learning-based prediction methods. To fully evaluate the performance of the energy efficiency, we choose three comparison algorithms from three aspects for comparative analysis. The first is the shortest path energy saving (SPES) algorithm based on Dijkstra, which is based on the greedy strategy to optimize network energy consumption without considering load balancing. The second is the *k*-paths energy efficiency selection (*k*-EES) algorithm, which requires multiple path calculations on the virtual topology and global topology and finally selects the path with the highest energy efficiency for routing. Note that energy consumption and load balancing are both considered in this comparison algorithm. The third is the classic load balancing algorithm Equal-Cost Multi-Path (ECMP). For fairness in comparison, we have also incorporated passive energy saving into ECMP (e-ECMP), i.e., when the link or node is idle, it will be turned OFF. Energy-saving ratio (esr) is defined as follows:

$$\text{esr} = \frac{P_O - P_\chi}{P_O} = \frac{P_O - \left(\sum_{e_{ij} \in E'} Pe_{ij} + \sum\limits_{v_j \in V'} Pv_j\right)}{P_O} \tag{39}$$

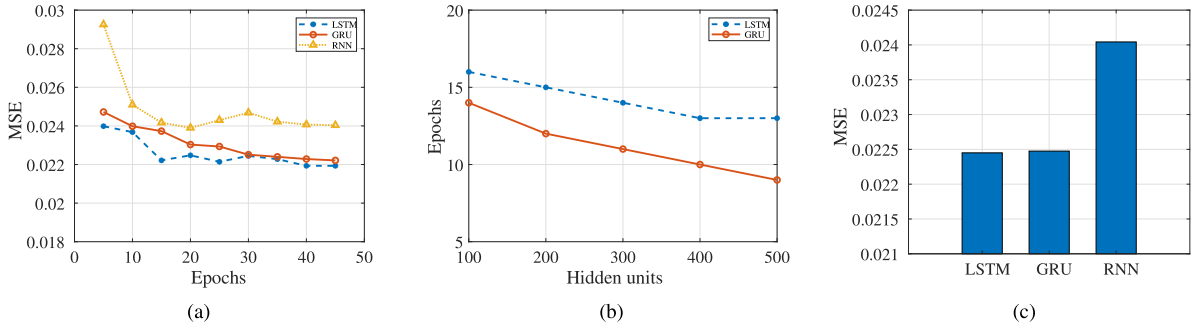


Fig. 3. (a) Prediction error under different epochs. (b) Training epochs under different hidden units. (c) Prediction performance of different methods.

where P_χ is the actual power consumption with energy-saving algorithms and P_O is the power consumption without energy saving.

In the experiment, deep learning models are trained on the basis of the Adam optimizer [35], which has been proven to have good training effect. We select traffic data of 750 traffic matrices, specifically, the first 0.7 * 750 are used for training and the last 0.3 * 750 are used for testing. The initial learning rate is 0.001, and the loss function is the MSE. time_step is set to 12, and the input dimension of flow data is (12, 144). The Adam optimizer adaptively adjusts the learning rate for model parameters and optimizes the parameters and minimizes the training cost of each layer. We first verify the MSE of the three methods under different epochs, as shown in Fig. 3(a). In the experiment, a three-layer network structure is adopted. With the increase of epoch, the MSE of all methods decreases and gradually reaches the convergence. RNN has the largest error, while LSTM and RNN are comparable. When the epoch is 40, the MSE of LSTM and GRU are around 0.022, and the RNN is about 0.024. Experiments show that the prediction performance of LSTM and GRU is better than RNN.

We evaluate the training epochs of LSTM and GRU with different hidden units under the certain loss. The training epochs are shown in Fig. 3(b). From the experimental results, we can see that as the hidden units increase, the training epochs decrease with low amplitude. GRU takes less training epochs than LSTM because GRU simplifies the gate structure inside the neural network and reduces model parameters. Experiments prove that GRU model has certain advantages in training when the accuracy of prediction is comparable to LSTM. Moreover, this advantage is more obvious with the increase of training data. Fig. 3(c) shows the average prediction MSE of the three methods under multiple experiments. GRU is roughly comparable to LSTM and significantly better than RNN. From Fig. 3(c), we conclude that the improved LSTM and GRU can fit network traffic better and achieve higher prediction accuracy.

Since the experimental results of the four mechanisms fluctuate and are difficult to compare directly, we use the cumulative distribution function (CDF) to compare the experimental results better. Here, Fig. 4(a) represents the curve of the original data over time, and Fig. 4(b) represents the CDF curve (the same as Figs. 5–9). The network power consumption is the sum of the power consumed by the switches and links that are turned ON.

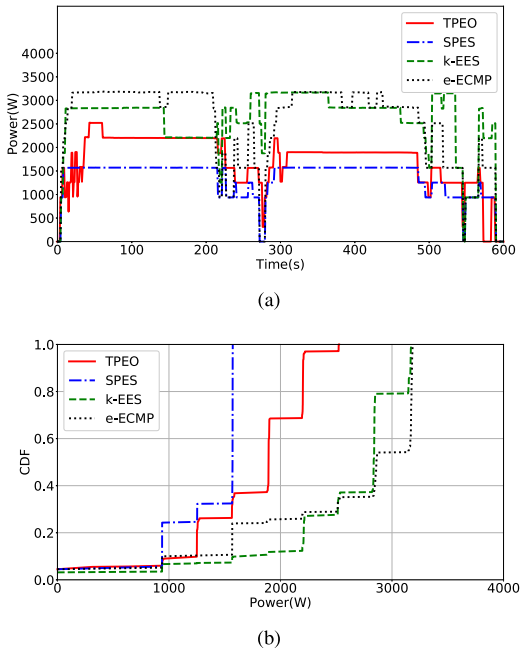


Fig. 4. (a) Power consumption. (b) CDF of power consumption.

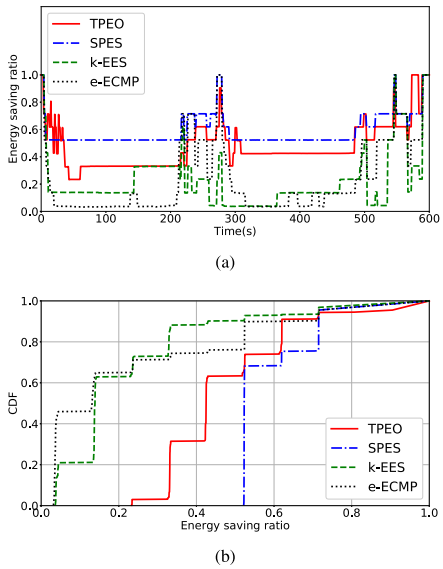


Fig. 5. (a) Energy-saving ratio. (b) CDF of energy-saving ratio.

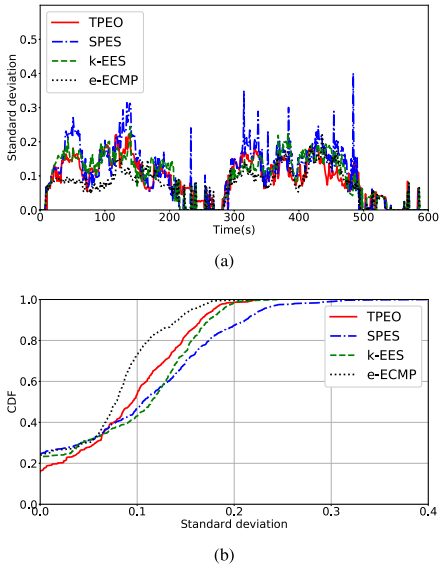


Fig. 6. (a) Standard deviation of link utilization. (b) CDF of standard deviation of link utilization.

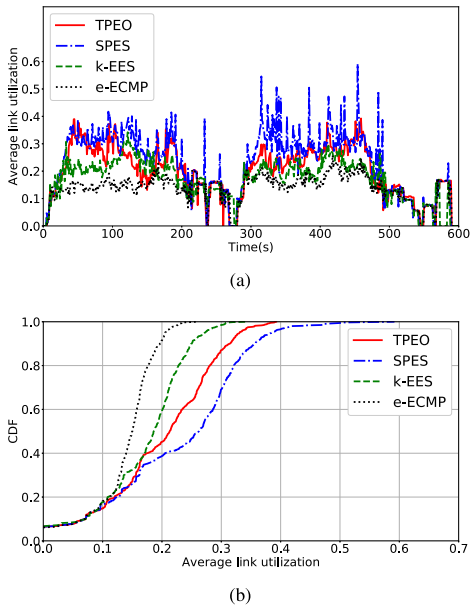


Fig. 7. (a) Average link utilization. (b) CDF of average link utilization.

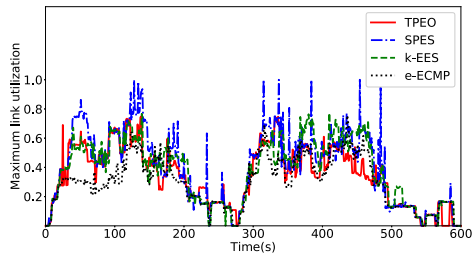
From the experimental results, we can see that as the network load changes, the state of switches and links also change, and the network power curve fluctuates continuously. In addition, the change of power consumption curve reflects the stability of the state of nodes and links to a certain extent.

Among the four algorithms, the average power consumption of TPEO, SPES, *k*-EES, and e-ECMP algorithms are 1721.335, 1349.240, 2560.679, and 2528.921 W, respectively. SPES uses the greedy energy-saving strategy based on the shortest path, which reduces energy consumption as far as possible. The TPEO algorithm proposed in this article is second only to the SPES and is significantly better than the other two comparison algorithms in general. The reason is that our algorithm implements real-time traffic prediction based on the GRU model and dynamically deploys energy-saving strategies to promptly optimize the network structure according to the changes of predicted network load. Furthermore, it disposes of the sleep/wake-up strategies based on the link/node priority to ensure the smooth transition of the network. The experimental results demonstrate that our algorithm achieves good energy-saving results while maintaining network stability.

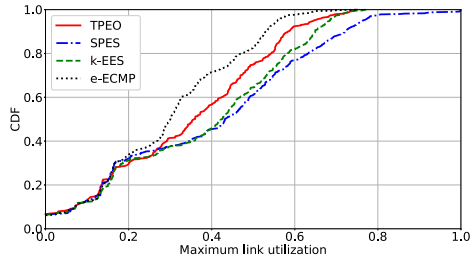
The energy-saving ratio esr is shown in Fig. 5(a) and (b). The esr is the ratio of the power saved by the energy-saving algorithm to the total power consumption without the energy-saving strategy. Experimental results show that the esr of all four algorithms are higher than 22%. The esr of our algorithm is as high as 47.71%, SPES is 58.99%, while that of *k*-EES and e-ECMP are 22.21% and 23.20%, respectively. The TPEO algorithm is second only to SPES and is obviously superior to the *k*-EES and e-ECMP. It shows that the TPEO algorithm can optimize the network structure promptly according to the dynamic changes of network load and improve the energy efficiency of the network.

The standard deviation of link utilization and the CDF are shown in Fig. 6(a) and (b), which characterize the load balance of the network. The standard deviation of link utilization of TPEO, e-ECMP, and *k*-EES are mainly distributed below 0.2. Specifically, from the cumulative distribution curve, e-ECMP mechanism accounts for 93.34% when the standard deviation is less than 0.15. Because in the e-ECMP algorithm, the controller selects a path through the hash according to the header information of the first data packet, which achieves good load balancing. In contrast, the SPES performs the worst, repeatedly higher than 0.25, and is extremely unstable. The reason is that the SPES does not take network load into account when saving energy. The TPEO algorithm is second to e-ECMP and better than SPES and *k*-EES. Experimental results show that our algorithm achieves not only good energy saving but also excellent load balancing performance. The main reason is that our algorithm considers both energy saving and load balancing when performing optimization strategies deployment and copes with the peak traffic by opening more devices. The average link utilization and its CDF of the four algorithms are demonstrated in Fig. 7(a) and (b), which reflect the ability to aggregate network traffic. Generally, the average link utilization shows an upward trend with the increase of total network traffic. Our algorithm is superior to the *k*-EES and e-ECMP on average link utilization. The reason is that our algorithm is based on the real-time predicted traffic data, which optimizes the network structure promptly according to the dynamic changes of network load and performs better on the aggregation effect of the flow.

Fig. 8(a) and (b) show the maximum link utilization and its CDF. The maximum link utilization of the TPEO is lower than 0.8, and the maximum link utilization less than 0.6 accounts for 92.35%, indicating that our algorithm realizes energy saving without link congestion. The throughput and its CDF are shown in Fig. 9. Since we use two days of traffic for simulation experiments, the four algorithms have little difference in terms of throughput. TPEO algorithm has the largest overall throughput and achieves the highest similarity to standard input.

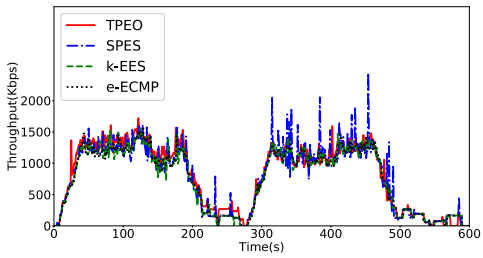


(a)

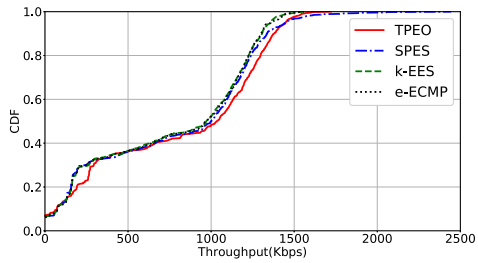


(b)

Fig. 8. (a) Maximum link utilization. (b) CDF of maximum link utilization.



(a)



(b)

Fig. 9. (a) Network throughput. (b) CDF of network throughput.

TABLE III
AVERAGE ROUTE HOPS AND AVERAGE ROUTE DELAY

Algorithm	Average route hops	Average delay(ms)
TPEO	3.15	112.54
SPES	3.00	115.10
k-EES	4.91	114.43
e-ECMP	4.18	111.87

The results of average route hops and average delay are shown in Table III. The average route hops of TPEO and SPES are 3.15 and 3.00, respectively, and they perform better than *k*-EES and e-ECMP. On the other hand, the average delay of e-ECMP and TPEO are 111.87 and 112.54 ms, respectively. Experiments show that TPEO achieves positive results in both average route hops and average delay. In actual network, the traffic load period of the network changes over a longer time interval. The period of traffic monitoring could be longer, and the caused network overhead will be further reduced. Meanwhile, the deployment time of the energy efficiency optimization strategies can be increased, and the algorithm is expected to have better performance in both energy saving and load balancing.

VII. CONCLUSION

In this article, we propose an energy efficiency algorithm in SDN based on traffic prediction by GRU neural network to optimize network energy efficiency and realize real-time load balancing. The algorithm coordinates the link resources with the dynamic changes of the network and deploys the sleep/wake-up strategies according to the subsequent traffic load to maximize the network energy efficiency. Simulation results show that our proposed algorithm can balance the network load and achieve a good energy-saving effect, and we expect the overall energy consumption to be reduced by around 47.71%. However, energy-saving strategies will result in relative network traffic concentration, which is susceptible to link failures. Thus, in the future works, we need to consider the reliability of the network while saving energy.

REFERENCES

[1] N. Surana and J. Mekie, "Energy efficient single-ended 6-t SRAM for multimedia applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 6, pp. 1023–1027, Jun. 2019.

[2] A. S. M. S. Hosen *et al.*, "A QoS-aware data collection protocol for LLNs in fog-enabled Internet of Things," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 430–444, Mar. 2020.

[3] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4188–4200, Jun. 2019.

[4] H. Dai, R. C. Wong, H. Wang, Z. Zheng, and A. V. Vasilakos, "Big data analytics for large-scale wireless networks: Challenges and opportunities," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 99:1–99:36, Sep. 2019.

[5] B. Addis, A. Capone, G. Carello, L. G. Gianoli, and B. Sansò, "Energy management through optimized routing and device powering for greener communication networks," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 313–325, Feb. 2014.

[6] "Cisco global cloud index: Forecast and methodology, 2016–2021 white paper," vol. 1, Cisco, San Jose, CA, USA, White Paper, Feb. 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/glo bal-cloud-index-gci/white-paper-c11-738085.html

[7] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey, "Recalibrating global data center energy-use estimates," *Science*, vol. 367, no. 6481, pp. 984–986, Feb. 2020.

[8] A. S. Andrae and T. Edler, "On global electricity usage of communication technology: Trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, Feb. 2015.

[9] W. Fisher, M. Suchara, and J. Rexford, "Greening backbone networks: Reducing energy consumption by shutting off cables in bundled links," in *Proc. 1st ACM SIGCOMM Workshop Green Netw.*, 2010, pp. 29–34.

[10] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices," in *Proc. 8th Int. Conf. Res. Netw.*, 2009, pp. 795–808.

[11] D. B. Rawat and S. R. Lenkala, "Software defined networking architecture, security and energy efficiency: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 325–346, Mar. 2017.

[12] B. G. Assefa and Ö. Özkasap, "A survey of energy efficiency in SDN: Software-based methods and optimization models," *J. Netw. Comput. Appl.*, vol. 137, pp. 127–143, Jul. 2019.

[13] R. Maaloul, L. Chaari, and B. Cousin, "Energy saving in carrier-grade networks: A survey," *Comput. Stand. Int.*, vol. 55, pp. 8–26, Jan. 2018.

[14] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures," *IEEE Commun. Surv. Tuts.*, vol. 13, no. 2, pp. 223–244, Jun. 2011.
[15] E. Rojas, R. D. Corin, S. Tamurejo, A. Beato, A. Schwabe, K. Phemius, and C. Guerrero, "Are we ready to drive software-defined networks? A comprehensive survey on management tools and techniques," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 27:1–27:35, Feb. 2018.
[16] J. Xie *et al.*, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surv. Tuts.*, vol. 21, no. 1, pp. 393–430, Feb. 2019.
[17] M. F. Tüysüz, Z. K. Ankarali, and D. Gözüpek, "A survey on energy efficiency in software defined networks," *Comput. Netw.*, vol. 113, pp. 188–204, Feb. 2017.
[18] H. Wang, Y. Li, D. Jin, P. Hui, and J. Wu, "Saving energy in partially deployed software defined networks," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1578–1592, May 2016.
[19] Z. M. Fadlullah *et al.*, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surv. Tuts.*, vol. 19, no. 4, pp. 2432–2455, May 2017.
[20] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *Proc. 31st Youth Acad. Annu. Conf. Chin. Assoc. Automat.*, 2016, pp. 324–328.
[21] X. Cao, Y. Zhong, Y. Zhou, J. Wang, C. Zhu, and W. Zhang, "Interactive temporal recurrent convolution network for traffic prediction in data centers," *IEEE Access*, vol. 6, pp. 5276–5289, Dec. 2017.
[22] R. Bolla *et al.*, "The green abstraction layer: A standard power-management interface for next-generation network devices," *IEEE Int. Comput.*, vol. 17, no. 2, pp. 82–86, Mar. 2013.
[23] R. Bolla, R. Bruschi, F. Davoli, and C. Lombardo, "Fine-grained energy-efficient consolidation in SDN networks and devices," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 2, pp. 132–145, Jun. 2015.
[24] B. R. Al-Kaseem and H. S. Al-Raweshidy, "SD-NFV as an energy efficient approach for M2M networks using cloud-based 6lowpan testbed," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1787–1797, Oct. 2017.
[25] S. Kao, D. Lee, T. Chen, and A. Wu, "Dynamically updatable ternary segmented aging bloom filter for openflow-compliant low-power packet processing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 1004–1017, Mar. 2018.
[26] X. Jia, Y. Jiang, Z. Guo, G. Shen, and L. Wang, "Intelligent path control for energy-saving in hybrid SDN networks," *Comput. Netw.*, vol. 131, pp. 65–76, Feb. 2018.
[27] S. Misra, S. Bera, M. P. Achuthananda, S. K. Pal, and M. S. Obaidat, "Situation-aware protocol switching in software-defined wireless sensors network systems," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2353–2360, Sep. 2018.
[28] M. Ndiaye, A. M. Abu-Mahfouz, and G. P. Hancke, "SDNMM—A generic SDN-based modular management system for wireless sensor networks," *IEEE Syst. J.*, vol. 14, no. 2, pp. 2347–2357, Jun. 2020.
[29] G. Xu, B. Dai, B. Huang, J. Yang, and S. Wen, "Bandwidth-aware energy efficient flow scheduling with SDN in data center networks," *Future Gener. Comput. Syst.*, vol. 68, pp. 163–174, 2017.
[30] A. Vishwanath, K. Hinton, R. W. A. Ayre, and R. S. Tucker, "Modeling energy consumption in high-capacity routers and switches," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 8, pp. 1524–1532, Aug. 2014.
[31] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "OpenTM: Traffic matrix estimator for openflow networks," in *Proc. 11th Int. Conf. Passive Act. Meas.*, vol. 6032, 2010, pp. 201–210.
[32] M. Polverini, A. Iacovazzi, A. Cianfrani, A. Baiocchi, and M. Listanti, "Traffic matrix estimation enhanced by sdns nodes in real network topology," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2015, pp. 300–305.
[33] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network monitoring in openflow software-defined networks," in *Proc. Netw. Oper. Manage. Symp.*, 2014, pp. 1–8.
[34] G. Dai, C. Ma, and X. Xu, "Short-term traffic flow prediction method for urban road sections based on space-time analysis and GRU," *IEEE Access*, vol. 7, pp. 143025–143035, 2019.
[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.
[36] Mininet. [Online]. Available: http://mininet.org/
[37] Ryu. [Online]. Available: https://ryu-sdn.org/
[38] Open vSwitch. [Online]. Available: https://www.openvswitch.org/
[39] TensorFlow. [Online]. Available: https://www.tensorflow.org/
[40] Keras. [Online]. Available: https://keras.io/
[41] H. Long, Y. Shen, M. Guo, and F. Tang, "LABERIO: Dynamic load-balanced routing in openflow-enabled networks," in *Proc. 27th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, 2013, pp. 290–297.
[42] AbileneTM. [Online]. Available: http://www.cs.utexas.edu/yzhang/research/AbileneTM/

Xiangyi Chen received the B.S. degree from Dalian Jiaotong University, Dalian, China, in 2017, and the M.S. degree in computer science and engineering in 2019 from Northeastern University, Shenyang, China, where she is currently working toward the Ph.D. degree in computer science and engineering.

Her research interests include software-defined networking, network function virtualization and mobile edge computing, etc.

Xingwei Wang received the B.S., M.S., and Ph.D. degrees in computer science from Northeastern University, Shenyang, China, in 1989, 1992, and 1998, respectively.

He is currently a Professor with the College of Computer Science and Engineering, Northeastern University. He has published more than 100 journal articles, books, and book chapters, and refereed conference papers. His research interests include cloud computing and future Internet.

Dr. Wang is the recipient of several best paper awards.

Bo Yi received the Ph.D. degree in computer application technology from the Northeastern University, Shenyang, China, in 2019.

He has authored or coauthored over ten journal articles. His research interests include routing and service function chain in SDN and NFV, etc. He is a Reviewer for IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, IEEE COMMUNICATIONS LETTERS, *Computer Networks*, *Journal of Network and Computer Applications*, etc.

Qiang He received the Ph.D. degree in computer application technology from the Northeastern University, Shenyang, China, in 2020.

He also worked with School of Computer Science and Technology, Nanyang Technical University, Singapore, as a Visiting Ph.D. Researcher from 2018 to 2019. He has published more than ten journal articles and conference papers. His research interests include social network analytic, machine learning, data mining, software defined networking, etc.

Min Huang received the B.S. degree in automatic instrument, the M.S. degree in systems engineering, and the Ph.D. degree in control theory from the Northeastern University, Shenyang, China in 1990, 1993, and 1999 respectively.

She is currently a Professor with the College of Information Science and Engineering, Northeastern University. Her research interests include modeling and optimization for logistics and supply chain system, etc. She has published more than 100 journal articles and books and has refereed many conference papers.