A Network Traffic Measurement Approach in Cloud-Edge SDN Networks

Liuwei Huo[1], Dingde Jiang[2(✉)], and Lisha Cheng[2]

[1] School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
[2] School of Astronautics and Aeronautic, University of Electronic Science and Technology of China, Chengdu 611731, China
jiangdd@uestc.edu.cn

Abstract. Edge computing is a supplement to cloud computing. It is deployed at the edge of the access network and is closer to where data is generated and used. In 5G and future networks, a large number of devices dynamically access the network and integrate them into cloud computing for deep processing and have high requirements for transfer rates and response time. However, network performance is the bottleneck of the collaboration between cloud computing and edge computing. Network traffic measurement is the core of network traffic management. In order to solve the problems of low utilization of network resources and high difficulty in network management, we study the problem of network traffic measurement in cloud edge computing networks based on software-defined networking (SDN). We propose a new cloud edge network traffic measurement method based on SDN. In this method, we extract statistical records coarse-grained from OpenFlow switches and use them to train an autoregressive moving average (ARMA) model. Use the ARMA model to make fine-grained predictions of network traffic. In order to reduce the estimation error, we propose to use optimization methods to optimize the estimation results. However, we found that the objective function is a very difficult NP-difficult problem, so we use a heuristic algorithm to quickly find the optimal solution. Finally, we repeat some simulations to evaluate the proposed method.

Keywords: Network traffic measurement · Software defined networking · Cloud-edge networks

1 Introduction

In the 5G and future networks, large capacity, low latency, and high dynamic will become the basic requirement of applications to the network. Improve network service capabilities by deploying cloud computing, edge computing, network slicing and network function virtualization (NFV), software-defined networking (SDN) and other new technologies in the access network and core network [1]. In the future, hundreds of millions of devices will need to connect to the network anytime and anywhere, which requires

H. Song and D. Jiang (Eds.): SIMUtools 2020, LNICST 369, pp. 204–214, 2021.
https://doi.org/10.1007/978-3-030-72792-5_19

high-performance cloud computing services to calculate and store massive amounts of data. However, as the number of access devices continues to increase, and in order to ensure that devices can access the network anytime and anywhere, a large number of access points must be deployed and corresponding access networks must be constructed. In this way, the network scale will be very large, and the access network will aggregate and process massive amounts of real-time data. The core networks will transmit and exchange a huge amount of data, which will be dozens of times the data volume of 4G/5G networks. The transmission network has become the bottleneck of data from service to cloud computing. For many delay-sensitive applications, the transmission from the terminal to cloud computing is intolerable. Edge computing is deployed in the network access network, and only one virtual resource pool contains multiple servers. In various terminal equipment, such as medical, industrial, and Internet of vehicles, many terminals and sensors are connected to the edge platform through the edge side. The pressure of resource shortage of edge computing is relatively high. Therefore, edge-cloud collaboration can effectively solve these problems.

Cloud-edge collaboration computing includes collaboration in computing resources, security policies, application management, and business management. To exchange the data between cloud computing and edge computing, the network must be flexibility and operability [2]. NFV utilizes IT (information technology) virtualization technology to perform network services on unified industrial standards, high-performance, large-capacity servers, or programmable switches to accelerate the development and update of new services. So, the network architecture must be changed for adopting these new applications. SDN decouples the control plane and forwarding plane in the traditional switch and centralizes the logical control plane to the SDN controller, so it enables the network to have functions such as flexibility, scalability, and programmability [3]. Network management has always been an important issue faced by operators and equipment vendors [4].

The volume of network traffic is the foundation of network management and often used for making decisions such as load balance, failure recovery, and anomaly detection [5]. Flow is the unit that is dispatched by switches and controllers in SDN. Flow that enters OpenFlow switch will match the flow entries in the flow table, and perform the corresponding actions [6]. In order to manage the cloud edge computing network and consider link load, quality of service (QoS) and network structure, the SDN controller needs to accurately understand the flow and link traffic.

Many measurement methods support flow-based measurement tasks such as sFlow, NetFlow, and SNMP (Simple Network Management Protocol) [7, 8]. sFlow and NetFlow are based on traffic sampling and packets of statistical methods, respectively. They are required the hardware supporting or the software that remote monitoring agent which runs in the network management server [7]. SNMP is a network management protocol that measures the network by sending probe packets. It has been widely used not only in a traditional network but also in the SDN network [8]. However, the SNMP protocol uses the polling method to collect the information of each switch in the network. The transmission of switch status information will consume a lot of bandwidth, and frequently reading switch information can cause network congestion.

In the SDN-based cloud edge computing network, we use pull-based methods for active network traffic measurement. This solution can not only measure flow flexibly and efficiently but can also be customized according to needs. Based on this, we extracted coarse-grained network traffic statistics from OpenFlow switches and used it to train the network traffic prediction model and infer other flows, and use optimization methods to estimate fine-grained network traffic optimization. The structure of this article is as follows. Section 1 is the summary. In Sect. 2, we describe a novel measurement architecture in the SDN-based cloud edge computing network, and use a traffic matrix to describe and analyze the scheme. In Sect. 3, we repeat the simulation to simulate the performance of the proposed scheme and compare the simulation results. The Sect. 4 is the conclusion.

2 Problem Statement

Cloud-edge computing networks are used for exchanging data between cloud computing and edge computing. Cloud computing will transmit applications to the edge computing platform, and edge computing transmits the raw data to the cloud computing platform for deep processing or storage. Then, the traffic between cloud computing and edge computing will have new characteristics. To this end, we study the cloud edge computing network traffic measurement under the SDN framework.

2.1 System Model

Network traffic measurement refers to selecting a representative grouping subset from the original traffic and tracking the characteristics of original traffic data through the grouping subset. With the increase of link capacity and the diversification of applications, huge network traffic measurement results are used for traffic recognition, transmission, storage, and analysis have brought huge pressure. To solve the problem of passive measurement of high-speed networks in SDN, we use the pull-based method and actively collect the statistics of Openflow-based switches for the high-speed cloud-edge computing network traffic measurement, which can reduce the use of measurement, storage, and processing under the condition of meeting the statistical accuracy of the problem.

In high-speed cloud-edge computing network traffic measurement, the implementation of active measurement is limited by technology and resources, and often requires a compromise between sampling rate and estimation accuracy. Coarse-grained traffic sampling can greatly reduce the processing load of the system and has better scalability, and can reflect the original flow characteristic parameters from the sample characteristic parameters, with certain measurement accuracy. In addition to the analysis of the flow characteristics, the sampling data is also widely used in the fields of traffic accounting, performance characteristic measurement, and abnormal detection. For flows in SDN, sampling methods are mainly used the flow sampling.

2.2 Traffic Matrix Construction

The network traffic matrix reflects the amount of traffic from the source to destination network nodes in the SDN-based cloud-edge computing network. Network engineering

and network management projects (such as congestion control, load balancing, network security, etc.) are based on the traffic matrix. Therefore, the flow matrix has very great practical significance. However, in the current actual network, due to the different support for the flow measurement function of the equipment produced by different network equipment manufacturers, it is very time-consuming and costly to obtain an accurate flow matrix through direct measurement. In contrast, the method of estimating the flow matrix by combining mathematical methods has become more feasible.

The network traffic from the origin host to the destination host at the time slot t is x_{ij}, where i is the origin host and j is the destination host. So, the traffic matrix can be written as

$$X = [x_{11}, x_{12}, \ldots, x_{1N}, x_{21}, x_{22}, \ldots, x_{2N}, \ldots, x_{N1}, \ldots, x_{NN}]^T \tag{1}$$

However, the transmission of flows in the network are aggregated on links, so the link load can reflect some features of the flows. With the routing matrix, we know the relationship between flows and links, so the relationship among traffic matrix, link load, and routing matrix can be described as:

$$Y = AX \tag{2}$$

where Y represent the traffic matrix of links, $\tilde{X}$ means the traffic matrix of flows, and A represents the relationship of routing in the cloud-edge computing network.

Estimating the cloud-edge computing network traffic is a typical inversion problem. As we all know, there is at least one stream between the source host and the destination host in the network, and there is a link that transmits multiple data streams. In the cloud edge network, a large number of applications are deployed on the cloud computing platform. When real-time sensitive applications request services, cloud computing will offload the applications to the edge computing platform. Since the number of flows in the cloud edge computing network is far greater than the number of links in the network, this means that there are unlimited solutions for traffic solutions. Then, we need to find some ways to search the optimized network traffic in the SDN-based cloud edge computing network.

In order to reduce the measurement overhead of the SDN-based cloud edge computing network, we perform coarse-grained sampling on the SDN-based cloud edge computing network traffic, forecast the fine-grained network traffic and estimate other network traffic, and then predicted network traffic through optimization.

2.3 ARMA Model

Edge computing and cloud computing exchange data through networks and the traffic can be described as a time sequence. The ARMA model (Auto-Regressive and Moving Average Model) is a method that is widely used for predicting the time sequence, it includes an autoregressive model (AR) and a moving average model (MA). ARMA can collectively reflect the characteristics of variance changes and has been widely used in long-term time series analysis and forecasting.

The traffic of flow can be formed as a random sequence over time. The dependence of the random sequence reflects the continuity of the original data in time. The AR model

can reflect the correlation of traffic in several adjacent time slots, so the traffic sequence $x_1, x_2, \ldots, x_t$ is

$$x_t = \sum_{i=1}^{p} \phi_i x_{t-i} + Z_t \tag{3}$$

where x_t represents the predicted value of the next time slot service sequence; Z_t represents the estimation error; $\phi_i (i = 1, \ldots, p)$ represents the autoregressive coefficient; p is the correlation order. The error Z_t is caused by white noise and can be expressed as a random sequence. Therefore, the MA model with random error is

$$Z_t = \gamma_t + \alpha_1 \gamma_{t-1} + \cdots + \alpha_i \gamma_{t-i} + \cdots + \alpha_q \gamma_{t-q} \tag{4}$$

where γ_t represents the Gaussian noise in the transmission links, the mean and variance of γ_t can be calculated as $E(\gamma_t) = 0$ and $E(\gamma_t^2) = \sigma^2$, respectively; The variable q is the moving average order; Variables $\alpha_j (j = 1, 2, \ldots, q)$ are the moving average coefficients. With the analysis above, the ARMA(p, q) is as follows:

$$\begin{aligned} x_t = \phi_1 x_{t-1} + \cdots + \phi_j x_{t-j} + \cdots + \phi_p x_{t-p} \\ + \gamma_t + \alpha_1 \gamma_{t-1} + \cdots + \alpha_i \gamma_{t-i} + \cdots + \alpha_q \gamma_{t-q} \end{aligned} \tag{5}$$

The ARMA(p, q) model can predict the flow traffic accurately by determining the order p and q accurately.

Use the pull scheme to extract statistics from the OpenFlow-based switch flow table. The ARMA model is trained on the front end of the measurement data and uses it to fill and reference the fine-grained traffic in the cloud edge computing network.

$$\hat{X} = \mathrm{ARMA}(X) \tag{6}$$

With the fine-grained network traffic that predicts by the ARMA model, we refer to the other flows with the constraints of link load. The fine-grained network traffic obtained by forecasting has an error with the actual network traffic, then we should optimize the forecasting result by utilizing the optimization method. In this process, we should measure the link load Y in SDN-based cloud-edge computing networks firstly. The objective of the flow traffic optimization is

$$f = \left\| Y - A\hat{X} \right\|_2 + \left\| \hat{X} \right\|_2 \tag{7}$$

where A means a routing matrix that can be obtained from the controller directly. Therefore, we use function (7) to construct a constrained target and use it to optimize the SDN-based cloud edge network traffic estimate. However, we find that the objective function is an NP-hard problem However, we find that the objective function is an NP-hard problem which cannot solve. To quickly approximate the optimal solution, we use a heuristic method to search for optimization results.

2.4 Artificial Fish Swarm Algorithm

The artificial fish swarm algorithm (AFSA) is a biomimetic optimization algorithm based on the intelligent behavior research of animal groups. It simulates the foraging behavior of fish swarm as they move toward nutritious areas in the waters. By simulating the behavior of a single fish, a global optimal value in the swarm is achieved through the local optimization of each fish. Artificial fish (AF) is an entity that abstracts and virtualizes real fish. It is composed of some characteristic data and a series of executable actions and can adjust its activities according to the information of the external environment. Each state of AF is located in the solution space and other AF solutions are the current environment. The next action of an AF will be affected by the solution space and its environment and will have a certain impact on other AF activities. The AF model uses the following methods to realize its virtual vision:

$$X_v = X + d_{visual} * Rand() \tag{8}$$

$$X_{next} = X + \frac{X_v - X}{\left\| X_v - \bar{X} \right\|} * S * Rand() \tag{9}$$

where X indicates the current state of AF; d_{visual} represents the visual distance of AF; *Rand()* is a random function that randomly produces a number in the range [0,1], with S being the step size.

A. Foraging Behavior

This mimics the foraging activity of fish. The AF considers the next swimming direction by sensing the amount or concentration of food within the sensing range of the water. By setting the current state of AF, and randomly selecting other states to swim within its sensing range. If the new state objective function of AF is greater than its current state, the state updated based on the new selection is closer to one step. Otherwise, please randomly select a new state to determine whether it is satisfied. AF X_i selects the new state X_j in its field of vision to update:

$$X_j = X_i + d_{Visual} * Rand() \tag{10}$$

Otherwise, X_i repeatedly calculate a new state X_j in its field of vision and determine whether the forward condition is satisfied. After repeatedly trying numbers, the forward condition is still not satisfied, and random behavior is performed.

B. Cluster Behavior

A large number of fish gathered together for food is a way of life in the evolution of fish. Each one will explore the current number of its adjacent individuals and calculate the center position of the fish cluster, and then compare the newly acquired center position target function with the current position target function. If the center position of the fish cluster is more crowded than the current position, the fish will from the current position swim towards the center position; otherwise, the fish will perform random foraging behavior.

AF X_i searches for the number of adjacent individuals n_f and the center position of the fish cluster X_c in its current visual field $d \leq d_{visual}$. If $X_c \big/ n_f < \delta X_i$ is satisfied, fish will be performed in the foraging process. We write the clustering behavior as

$$X_i^{t+1} = X_i^t + \frac{X_c - X_{it}^t}{\|X_c - X_i\|} * S * Rand() \tag{11}$$

where X_c represents the center position of the AF cluster.

C. Rear-ending Behavior

When some find food, other fish will follow them to swim, causing fish to gather the food. The AF swim towards the optimal position of nearby fish. If a large number of fish gather together, it will cause AF to be overcrowded at the optimization point. Therefore, we introduce a threshold variable to prevent the AF from over clustering. If the optimization point is not very crowded, the AF will swim from the current position towards the optimal AF one step, otherwise, the AF swims a step randomly.

AF X_i looking for the optimal individual X_j in the partner of the current field of vision ($d_{ij} < d_{Visual}$). If $X_j/n_f > \delta X_i$ means that the optimal individual is not too crowded nearby, then X_i moves towards the partner, otherwise foraging behavior.

$$X_i^{t+1} = X_i^t + \frac{X_j - X_{it}^t}{\|X_j - X_i\|} * S * Rand() \tag{12}$$

D. Random Behavior

It is the foraging behavior of fish, which means that AF moves randomly within the visual field. When a fish find the food, it moves quickly in the direction of increasing food. The algorithm describes that the AF X_i moves towards the food a step and updates the current state:

$$X_i^{t+1} = X_i^t + Visual * Rand() \tag{13}$$

E. Bulletin board

The bulletin board records the best place to update the status of the fish cluster. After each iteration, each AF compares its current state with the best state recorded on the bulletin board. If the objective function value of the current state of the fish is higher than the objective function value of the bulletin board state, the state of the bulletin board update as its status; otherwise, the state of the bulletin board will remain unchanged. When the iterative process of the algorithm ends, the value recorded on the bulletin board is the best solution for the entire cluster of fish . The traffic in the SDN-based cloud edge computing network is AF, and the behavior evaluation reflects the autonomous behavior of AF.on the bulletin board is the best solution for the entire cluster of fish.

3 Analysis of Simulation Result

3.1 Simulation Environment

We evaluate the proposed network traffic measure and prediction method by implementing some simulations with the Ryu, Mininet, and Docker. We use python programming

to implement functional modules and install them into the Ryu controller. Then, we use the Mininet to create the network topology and use the Open vSwitch to simulate the OpenFlow switches and create some hosts in the Mininet topology, and also use the docker to create the hosts and mount them on to the OpenFlow switch. Iperf is a software that can generate traffic and fill them into the access switches from the origin host and catching them at the destination host, then we send data from the origin host to the destination host. We used docker to create hosts H5, H8, H10 and H11, and transmit big packaged data from them to other hosts. The network traffic between two hosts is Fig. 1 shows.

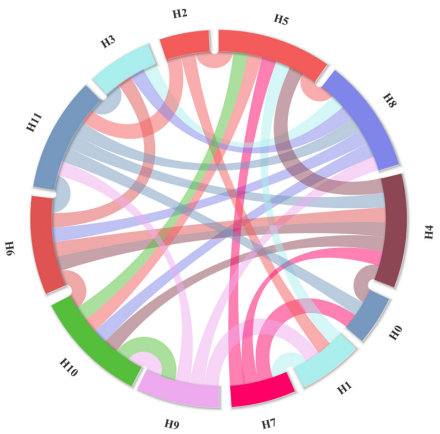

Fig. 1. Flows between two hosts in the SDN-based cloud-edge computing network.

To intuitively display and analyze the performance of the proposed scheme, we have introduced absolute error (AE) and relative error (RE) indicators. The AE and RE can be expressed as

$$AE_i = \left| x_i - \hat{x}_i \right| \tag{14}$$

$$RE_i = \left| x_i - \hat{x}_i \right| \Big/ \hat{x}_i \tag{15}$$

3.2 Simulation Evaluation

In this article, we choose two flows f1 and f2 as examples for discussion, as shown in Fig. 2. Therefore, in this scheme, we use ARMA and AFSA models to estimate and optimize the traffic in the cloud-side network. In the figure, we use ARMA-ASFA stands for the scheme proposed in this article. Then, compare and analyze with ARMA and Principal Component Analysis (PCA) methods.

Figures 3 and 4 respectively show the AE and RE of the measured results under different measurement methods. It can be seen from Fig. 3 that the AE of the ARMA-AFSA mentioned in this article is smaller than the AE of the measurement results of

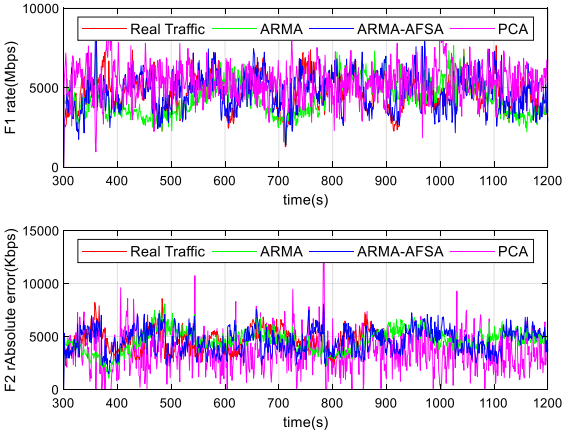

Fig. 2. Measurement results of network traffic.

ARMA and PCA. The flow in Figs. 3 and 4 fluctuate greatly. The AEs of ARMA-AFSA, ARMA and PCA all exceed 1000 bps. The reason is that the flow has random characteristics and the estimation results cannot eliminate Gaussian white noise in the transmission channel. The measured value trends of different flow rates are similar, and the relative error of the ARMA-AFSA and ARMA measurement methods is less than 0.3. Regardless of AE or RE, PCA has the largest error among these three methods. We also compare the proposed method with the RE of the PCA method and the ARMA method. The RE of the proposed scheme ARMA-AFSA is significantly smaller than the PCA and ARMA methods in Fig. 4.

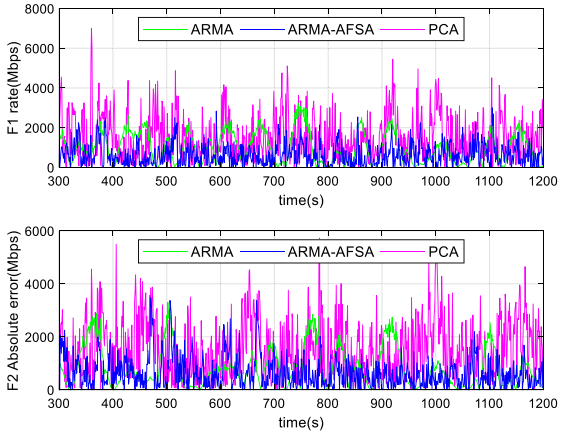

Fig. 3. Measurement results of network traffic.

In Fig. 5, we compare the CDF of RE of three network traffic estimation and measurement schemes. We have noticed that the RE of the proposed measurement scheme is

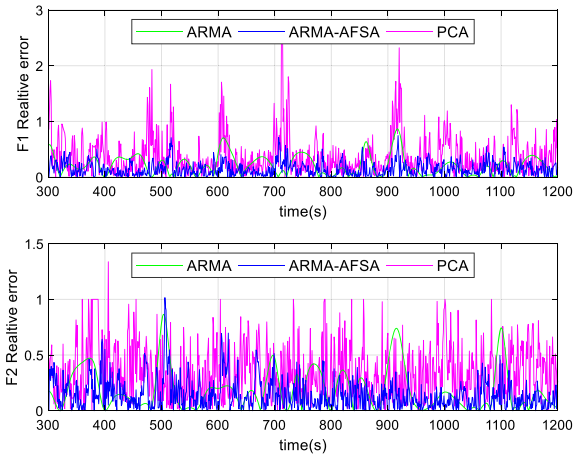

Fig. 4. The RE of different methods.

better than the CDF of the PCA method and the ARMA method. This is mainly because we use coarse-grained measurement results to help the ARMA model obtain the fine-grained measurement result, and use the AFSA method to reduce estimation errors. The optimization for network traffic estimation results is effective as shown in Fig. 5 directly.

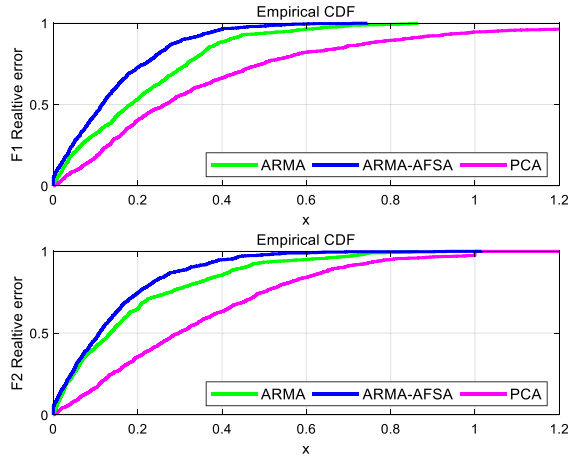

Fig. 5. The CDF of the relative error for different methods.

4 Conclusions

In the SDN-based cloud-edge computing network, the fine-grained network traffic measurement result is significant for network management. Since the network between cloud

computing and edge computing must flexible and efficient, we introduce SDN into the cloud-edge computing network and propose a novel measurement method. We use pull-based mechanisms to extract the coarse-grained statistics record of flows and links in the flow table in OpenFlow-based switches. Then, we construct an ARMA model and use the front measurement data to train it , and use the proposed ARMA model to forecast and optimize the estimation results to reduce measurement errors. Finally, we repeat some simulations to verify the proposed method in the SDN-based cloud-edge computing network.

Acknowledgments. The work was supported in part by the National Natural Science Foundation of China (No. 61571104), the Sichuan Science and Technology Program (No. 2018JY0539), the Fundamental Research Funds for the Central Universities (No. ZYGX2017KYQD170), the Key projects of the Sichuan Provincial Education Department (No. 18ZA0219), the CERNET Innovation Project (No. NGII20190111), the Fund Project (Nos. 61403110405, 315075802), and the Innovation Funding (No. 2018510007000134). The authors wish to thank the reviewers for their helpful comments.

References

1. Long, Q., Chen, Y., Zhang, H., et al.: Software defined 5G and 6G networks: a survey. Mobile Netw. Appl. (5), 1–21 (2019)
2. Jain, A., Lopez-aguilera, E., Demirkol, I.: Are mobility management solutions ready for 5G and beyond? Comput. Commun. **161**, 50–75 (2020)
3. Oh, B., Vural, S., Wang, N., et al.: Priority-based flow control for dynamic and reliable flow management in the SDN network. IEEE Trans. Netw. Serv. Manage. **15**(4), 1720–1732 (2018)
4. Tian, Y., Chen, W., Lea, C.: An SDN-based traffic matrix estimation framework. IEEE Trans. Netw. Serv. Manage. **15**(4), 1435–1445 (2018)
5. Liu, Z., Wang, Z., Yin, X., et al.: Traffic matrix prediction based on deep learning for dynamic traffic engineering. In: Proceedings of IEEE Symposium on Computers and Communications (ISCC), July 2019, pp. 1–7
6. Huo, L., Jiang, D., Qi, S., et al.: An AI-based adaptive cognitive modeling and measurement method of network traffic for EIS. Mobile Netw. Appl. **12**, 1–12 (2019)
7. Suarez-varela, J., Barlet-ros, P.: Flow monitoring in software-defined networks: finding the accuracy/ performance tradeoffs. Comput. Netw. **135**, 289–301 (2018)
8. Karakus, M., Durresi, A.: An economic framework for analysis of network architectures: SDN and MPLS cases. J. Netw. Comput. Appl. **136**, 132–146 (2019)
9. Jiang, D., Huo, L., Li, Y.: Fine-granularity inference and estimations to network traffic for SDN. PLoS ONE **13**(5), 1–23 (2018)
10. Liu, C., Malboubi, A., Chuah, C.: OpenMeasure: adaptive flow measurement and inference with online learning in SDN. In: Proceedings of INFOCOM'16, pp. 47–52 (2016)
11. Shu, Z., Wan, J., Wang, S., et al.: Traffic engineering in software-defined networking: measurement and management. IEEE Access **4**, 3246–3256 (2016)