Research on Load Balancing for Software Defined Cloud-Fog Network in Real-Time Mobile Face Recognition

Chenhua Shi, Zhiyuan Ren(✉), and Xiuli He

The State Key Laboratory of ISN in School of Telecommunications Engineering,
Xidian University, Xi'an, Shaanxi, China
{chshi,zyren,xlhe}@s-an.org

Abstract. The real-time camera-equipped mobile devices have been widely researched recently. And cloud computing has been used to support those applications. However, the high communication latency and unstable connections between cloud and users influence the Quality of Service (QoS). To address the problem, we integrate fog computing and Software Defined Network (SDN) to the current architecture. Fog computing pushes the computation and storage resources to the network edge, which can efficiently reduce the latency and enable mobility support. While SDN offers flexible centralized control and global knowledge to the network. For applying the software defined cloud-fog network (SDC-FN) architecture in the real-time mobile face recognition scenario effectively, we propose leveraging the SDN centralized control and fireworks algorithm (FWA) to solve the load balancing problem in the SDC-FN. The simulation results demonstrate that the SDN-based FWA could decrease the latency remarkably and improve the QoS in the SDC-FN architecture.

Keywords: Mobile face recognition · Cloud computing · Fog computing · Cloud-fog network · Software Defined Network · Load balancing

1 Introduction

With the rapid popularization of mobile terminals, it is useful and convenient to detect and recognize face on smart phones, tablets or laptops, which causes numerous novel applications based on face recognition on mobile devices, such as pay-with-your-face, photo tagging, face login and etc.

The face recognition applications on mobile devices require real-time response time and mobility support. However, lots of face information needs to be processed during the course of recognition. Thus, it is difficult for the resource constrained mobile devices to process computationally intensive real-time recognition tasks. Offloading the real-time face recognition tasks to the cloud computing platform is naturally regarded as a competitive method to tackle such limitation.

In the cloud-based network architecture, cloud servers provide powerful computation and storage capacity for the face recognition applications. But there still remains several challenges. It takes a relatively long time for users to send images to the cloud since the cloud is far from end users. Furthermore, as more and more intelligent services are supported by cloud computing, the load of the cloud is heavier, which leads to a poor robustness. Therefore, the cloud-based network architecture would not satisfy the latency requirement of real-time face recognition well.

To overcome the above problem, in this paper we propose a novel software defined cloud-fog network architecture which integrates fog computing and Software Defined Network (SDN) to the cloud-based architecture. The employment of SDN can ease the control of the network, increase network scalability and provide global knowledge to the network [1]. To support the real-time mobile face recognition service, we introduce fog computing. Fog is considered as a cloud close to the end users, which offers computation and data resources at the network edge, and thus enables a new breed of services that require low latency, mobility support and geo-distribution [2]. However, the fog network usually consists of a large number of distributed resource-poor devices, and a single fog device can't efficiently process numerous tasks. Therefore, it's necessary to execute distributed computing in fog network.

Load balancing is one of the key technologies of the distributed computing. Balancing the load according to an effective load balancing strategy can reduce the response time remarkably. As the load on the cloud increases tremendously, lots of works have been researched to balance the load of cloud computing [3, 4]. Although fog is usually considered as a local cloud, the load balancing strategies of the cloud computing can't be applied to the fog network directly since the fog network is heterogeneous and dynamic. Moreover, very few literatures concern about the load balancing of task processing in the fog network. Most existing researches mainly focus on the applications, resource allocation and energy management [5–7]. Therefore, we investigate the efficient load balancing policy in the software defined cloud-fog network (SDC-FN) to decrease the latency.

The main contributions of this paper are summarized as follows:

(1) We integrate fog computing and SDN to the cloud-based mobile face recognition architecture to solve the latency problem.
(2) We formulate the load balancing in SDC-FN as an optimization problem.
(3) We propose applying fireworks algorithm (FWA) based on SDN centralized control to solve the load balancing problem.

The rest of the paper is structured as follows. In Sect. 2, we introduce the SDC-FN architecture; In Sect. 3, we formulate a theoretical model of the load balancing problem in the SDC-FN and propose applying fireworks algorithm (FWA) based on SDN centralized control to solve the load balancing problem; Our simulation results are described in Sect. 4. Finally, we conclude our work in Sect. 5.

2 SDC-FN Architecture

When users use the face recognition applications, they take face photos with mobile terminals, and then the applications send the photo information to the processing center to perform the following steps: face detection, projection and a database search for getting the recognition results. In SDC-FN, in order to decrease the response time, we introduce fog network to perform the preprocess operations including face

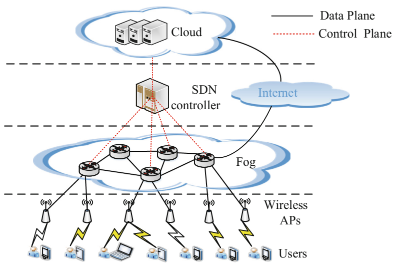

Fig. 1. SDC-FN architecture

detection and projection. Meanwhile, SDN is necessary for its centralized control. The overall architecture is shown in Fig. 1.

The architecture comprises of infrastructure layer, fog computing layer, control layer, and cloud computing layer.

The infrastructure layer consists of mobile terminals and wireless Access Points (Aps). Mobile terminals connect to the APs through one hop wireless link. APs are located on the network edge, which can be deployed in high density. Meanwhile, APs run the OpenFlow protocol, which are responsible for forwarding the received data. And the forwarding rules are formulated by SDN controller.

The fog computing layer is composed of edge network devices (e.g., routers, switches) whose computing and storage capability are limited. Fog devices are OpenFlow-enabled, which not only interact with SDN controller, but also collaborate with APs to forward data rapidly. Moreover, the face recognition task can be preprocessed by fog devices, such as face detection and projection, thereby decrease the communication latency and alleviate the burden on the cloud. Since the preprocessing operations are computationally-intensive operations, it will lead to long latency that numerous preprocessing tasks are handled by a single fog device. Therefore, it is essential to execute distributed computing to balance the load.

The control layer includes SDN controller. OpenFlow-enabled SDN controller controls the SDC-FN in a centralized way and it can collect the global knowledge of the topology by interacting with fog devices and APs. Moreover, we run the load balancing algorithm on the controller to develop an optimal load balancing strategy.

The cloud computing layer consists of cloud servers. Cloud servers utilize their huge storage capacity to store a large quantity of facial information and set up a face database. The facial feature information extracted by fog devices is delivered to the cloud to match with the known faces in the database.

3 FWA-Based Load Balancing Algorithm in SDC-FN

3.1 Theoretical Model

We consider the fog network with k fog devices. The network topology is illustrated in Fig. 2.

We can abstract the above topology as a weighted undirected graph $G = (\boldsymbol{V}, \boldsymbol{E})$ with vertex set $\boldsymbol{V}$ and edge set $\boldsymbol{E}$, as shown in Fig. 3.

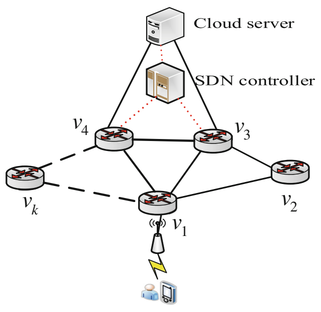

Fig. 2. SDC-FN topology

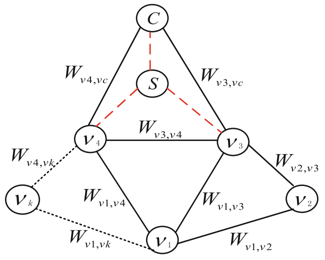

Fig. 3. The weighted undirected graph

In Fig. 3, $\boldsymbol{V} = \{v_1, v_2, \cdots, v_k, \mathrm{S}, \mathrm{C}\}$, where vertex v_i denotes the fog device, S and C represent the SDN controller and the cloud respectively. We denote the computing capacity of fog device v_i as c_{vi}, and the computing capacity of the cloud sever is c_c. In edge set $\boldsymbol{E} = \left\{e_{v_1 v_2}, \cdots, e_{v_i v_j}, \cdots, e_{v_{k-1} v_k}, e_{v_3 v_c}, e_{v_4 v_c}\right\}$, each edge represents a communication links between nodes, and the weight of each edge $e_{vi,vj}$, i.e., $w_{vi,vj}$ represents the communication latency between nodes v_i and v_j. During the course of face recognition, the recognition tasks, i.e., *Task* received by fog device v_i can be divided into many small subtasks $Task_i$ firstly, which satisfies the condition $Task_i = \delta_i Task$, where δ_i is the portion of the subtask in the total task. Secondly, the subtasks are allocated to appropriate fog devices to perform preprocessing operations. Finally, the results of preprocessing $Task_{pre}$ will be transmitted to the cloud for the final recognition results and the results will be sent back to end users. Therefore, the total processing time t of the task in SDC-FN can be expressed as:

$$t = \max\left\{\frac{\delta_i Task}{c_{v_i}} + w_{v_i,v_j}\ m_{v_i,v_j}\right\} + \frac{Task_{pre}}{c_c} + w_{v_i,c} \tag{1}$$

where $\delta_i Task/c_{vi}$ denotes the computation time of the subtask $Task_i$ on fog device v_i, $w_{vi,vj}$ is the communication latency between fog devices v_i and v_j, $m_{vi,vj}$ denotes whether there is a subtask allocation relationship between fog devices v_i and v_j. When $m_{vi,vj} = 1$, there exists the relationship; when $m_{vi,vj} = 0$, the relationship doesn't exist. The $Task_{pre}/c_c$ part is the computation time of the task $Task_{pre}$ on the cloud, and $w_{vi,c}$ represents the communication latency between fog device v_i and cloud.

For achieving the minimum task processing latency t, we must find a group of optimal δ_i.In summary, the problem can be formulated as:

$$\min\left\{\max\left[\frac{\delta_i Task}{c_{v_i}} + w_{v_i,v_j} m_{v_i,v_j}\right] + \frac{Task_{pre}}{c_c} + w_{v_i,c}\right\} \quad i,j = 1, 2, \cdots, k. \tag{2}$$

$$s.t.\, m_{v_i,v_j} = \begin{cases} 1, & \delta_i \neq 0 \\ 0, & \delta_i = 0 \end{cases} . \qquad \sum_{i=1}^{k} \delta_i = 1 \tag{3}$$

3.2 SDN-Based FWA Algorithm

In the model of the load balancing problem in Sect. 3.1, it is necessary to find a set of optimal load distribution coefficients δi to obtain the minimum latency. In SDC-FN, the subtask processed on each fog device is $Task_i = \delta_i Task$. Accordingly, the subtasks on k fog devices form a k dimension vector $\boldsymbol{TA} = (Task_1, Task_2, \ldots, Task_k)^T$. Assuming the tasks are received by fog device v_1, from Eq. (1), the total latency t can be expressed as:

$$t(\boldsymbol{TA}) = \max\left\{\frac{Task_1}{c_{v_1}} + w_{v_1,v_1} m_{v_1,v_1}, \cdots, \frac{Task_k}{c_{v_k}} + w_{v_1,v_k} m_{v_1,v_k}\right\} + \frac{Task_{pre}}{c_c} + w_{v_1,v_c} m_{v_1,v_c} . \tag{4}$$

The resolution of δ_i can be converted into the resolution of vector $\boldsymbol{TA}$, which could be formulated as the following optimization problem:

$$\min\{t(\boldsymbol{TA})\}, \boldsymbol{TA} \in I . \tag{5}$$

$$s.t.\ \boldsymbol{TA}(i) \geq 0 \qquad \sum_{i=1}^{k} \boldsymbol{TA}(i) = Task \cdot \tag{6}$$

And the solution space I is:

$$I = \prod_{i=1}^{k} [Task_{i\min}, Task_{i\max}] = \prod_{i=1}^{k} [0, Task] . \tag{7}$$

In this paper, we introduce fireworks algorithm (FWA) to solve the load balancing problem, namely the above optimization problem. FWA is one of the latest swarm intelligence optimization problem proposed by Tan and Zhu [8]. Although there has been few works about the FWA's implementation, the results show that it has exhibited promising performance in dealing with various optimization problems [9, 10].

The steps of leveraging FWA to resolve the load balancing problem, i.e., Eq. (5), are shown as follows:

(1) The fireworks $\{\boldsymbol{TA}_i\}_{i=1}^{N}$ should be randomly initialized in the solution space, where N denotes the number of fireworks. The position of each firework is $\boldsymbol{TA}_i = (Task_1(i), Task_2(i), \cdots, Task_k(i))^T$.

(2) Computing the fitness value of each firework according to the optimization objective function $t(\boldsymbol{TA})$. The explosion amplitude A_i and the number S_i of explosion sparks for each fireworks $\boldsymbol{TA}_i$ can be defined as: [8]

$$A_i = A \times \frac{t(\boldsymbol{TA}_i) - t_{\min} + \varepsilon}{\sum_{i=1}^{N} \left(t(\boldsymbol{TA}_i) - t_{\min}\right) + \varepsilon}. \tag{8}$$

$$S_i = M \times \frac{t_{\max} - t(\boldsymbol{TA}_i) + \varepsilon}{\sum_{i=1}^{N} \left(t_{\max} - t(\boldsymbol{TA}_i)\right) + \varepsilon}. \tag{9}$$

where A and M are two constants for controlling the maximum value of the explosion amplitude and the number of explosion sparks, $t_{max} = max(t(\boldsymbol{TA}_i))$ and $t_{min} = min((t(\boldsymbol{TA}_i))$ $(i = 1,2,\ldots,N)$ represent the maximum and minimum fitness value in the current fireworks population respectively, and ε is the machine epsilon to avoid zero-division-error. In order to avoid the overwhelming effects of the better fireworks, it's necessary to limit the number of sparks S_i: [8]

$$\hat{S}_1 = \begin{cases} round(a * M), \mathrm{S}_i < aM \\ round(\mathrm{b} * M), \mathrm{S}_i > bM, a < b < 1 \\ round(\mathrm{S}_i), \text{ otherwise} \end{cases} \tag{10}$$

where a, b are two constants.

(3) Generating explosion sparks according to the calculated number of the sparks S_i and the explosion amplitude A_i. When an explosion occurs, there exists a random offset value in $[-A_i, A_i]$, which is added to z dimensions randomly chosen from $\boldsymbol{TA}_i$. The chosen z dimensions are calculated as: [8]

$$z = round(d \times U(0,1)) \tag{11}$$

where d represents the dimension of $\boldsymbol{TA}_i$, $U(0,1)$ is a random number uniformly distributed between 0 and 1. Each dimension $k \in \{1,2\ldots,z\}$ of the generated explosion sparks $\boldsymbol{T\hat{A}}_i$ can be expressed as: [8]

$$\boldsymbol{T\hat{A}}_{ik} = \boldsymbol{TA}_{ik} + A_i \times U(-1,1) \tag{12}$$

where $U(-1,1)$ is a random number uniformly distributed between −1 and 1
If $\boldsymbol{T\hat{A}}_i$ is out of the bound of the solution space on the dimension k, it will be mapped to a new location according to the equal: [8]

$$\boldsymbol{T\hat{A}}_{ik} = \boldsymbol{TA}_{LB,k} + \left|\boldsymbol{T\hat{A}}_i\right| \% \left(\boldsymbol{TA}_{UB,k} - \boldsymbol{TA}_{LB,k}\right) \tag{13}$$

where $\boldsymbol{TA}_{UB,k}$ and $\boldsymbol{TA}_{LB,k}$ denote the upper and lower boundary of the solution space on the dimension k, respectively.

(4) Generating Gaussian mutation sparks to maintain the diversity of sparks. FWA randomly selects *Mg* fireworks in the fireworks population and chooses *z* dimensions randomly according to Eq. (11) in each firework of these *Mg* fireworks to perform Gaussian mutation. (*Mg* is a constant to control the number of Gaussian sparks). Each dimension $k \in \{1,2\ldots,z\}$ of the generated Gaussian sparks $\hat{\boldsymbol{TA}}_i$ can be calculated as: [8]

$$\hat{\boldsymbol{TA}}_{ik} = \boldsymbol{TA}_{ik} \times N(1,1). \tag{14}$$

where *N*(1,1) represents the Gauss distribution with mean 1, variance 1. Similarly, if $\hat{\boldsymbol{TA}}_i$ is out of the bound of the solution space on the dimension *k*, it will be mapped to a new location according to Eq. (13).

5) Selecting *N* individuals as a new fireworks population from the current generation of fireworks, explosion sparks and Gaussian sparks to enter into the next generation. The individual with minimum fitness value is deterministically transmitted to the next generation. In order to maintain the diversity, the remaining *N−1* individuals are selected based on Roulette Wheel Selection. For individual $\boldsymbol{TA}_i$, the selection probability $P(\boldsymbol{TA}_i)$ is calculated as: [8]

$$P(\boldsymbol{TA}_i) = \frac{R(\boldsymbol{TA}_i)}{\sum\limits_{\mathbf{TA}_j \in K} \boldsymbol{TA}_j} \tag{15}$$

$$R(\boldsymbol{TA}_i) = \sum_{\boldsymbol{TA}_j \in K} d(\boldsymbol{TA}_i - \boldsymbol{TA}_j) = \sum_{\boldsymbol{TA}_j \in K} \left\| \boldsymbol{TA}_i - \boldsymbol{TA}_j \right\| \tag{16}$$

where *K* is the set of all current individuals including both fireworks and sparks, $R\ (\boldsymbol{TA}_i)$ is the distance between individual $\boldsymbol{TA}_i$ and other individuals.

(6) Repeating step 2–5 until the end termination condition is reached.
In the above analysis of the FWA-based load balancing strategy, SDN controller collects the information of all nodes, formulates an optimal load balancing strategy by running the FWA and sends flow tables including the strategy to fog devices.

4 Simulation Results

We consider the scenario with 10 fog devices. In the real network circumstance, the communication latency $w_{vi,vj}$ between the nodes mainly includes uplink transmission latency, downlink transmission latency and other latency. Other latency includes propagation latency and queuing latency. Some important parameters of SDC-FN and FWA in the simulation are summarized in Tables 1 and 2 respectively, referring to [9, 11]. The task loads in the experiment are simulation setting.All the simulating results are obtained by MATLAB and they are the means of many repeated experiments.

Table 1. The related parameters of SDC-FN

Parameter	SDC-FN devices										
Type	C	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
c_c/c_{vi} (Gbps)	10	1	2	3	0.4	0.5	0.6	0.6	0.7	0.5	1
Uplink bandwith (Mbps)	2	84	86	71	80	83	90	86	89	87	79
Downlink bandwith (Mbps)	1.8	99	100	98	101	96	104	99	105	102	97
Other latency (ms)	10	1	1	1.2	1.1	1.2	1	1.3	1	1.3	1

Table 2. The related parameters of FWA algorithm

Parameters	Values
Number of fireworks N	20
Maximum value of the explosion amplitude A	30
Number of sparks M	64
Constant parameter a	0.04
Constant parameter b	0.8
Number of Gaussian sparks Mg	20

4.1 Latency Performance Comparison Between the Cloud-Based Architecture and SDC-FN

In the simulation, we compare the SDC-FN based on FWA load balancing algorithm with the cloud-based architecture to evaluate the latency performance of the SDC-FN.

The comparison result is shown in Fig. 4. When the workload is lower than 1 Gb, the recognition task processing latency in SDC-FN is lower than in the cloud-based architecture, but their latency values have little difference. This is because in the cloud-based architecture, the computing capacity of the cloud severs is more powerful, and transmitting a small amount of tasks to the cloud will not generate high communication latency. However, with the increase of task, the latency

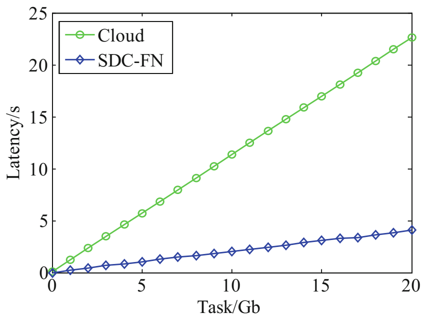

Fig. 4. Latency performance comparison between the cloud-based architecture and SDC-FN

in the cloud-based architecture is dramatically higher than that in SDC-FN because of the increasing transmission latency. Compared with the cloud-based architecture, the reason why SDC-FN shows good latency performance is that the cloud is far from the users and there is limited bandwidth while the fog is proximity to end users and the communication bandwidth is higher. Therefore, the SDC-FN architecture can efficiently reduce the latency to improve the QoS.

4.2 Latency Performance Comparison of Multiple Load Balancing Algorithms

To validate the high efficiency of the FWA-based load balancing algorithm in reducing latency in SDC-FN, we compare it with the PSO-CO [12], Weighted Round Robin (WRR) [13] and Pick-KX load balancing algorithm [14].

Figure 5 shows the simulation results of four load balancing algorithms in SDC-FN. With the increase of the task quantity, the FWA-based algorithm obtained lower latency than other three algorithms. On one hand, in FWA, the fireworks with low fitness generate more sparks in a small range, which has a strong local search capability for the location of the fireworks. Conversely, the high fitness fireworks create less sparks in a large scale, which has a certain global search capability. Thus, the FWA-based algorithm can achieve a better global load balancing strategy. On the other hand, the WRR and Pick-KX algorithm don’t consider the communication latency when balancing the load, and the PSO-CO may fall into the local optimum, thereby they can’t formulate a good load balancing strategy to reduce latency. When the task quantity is 20 Gb, the delay performance of FWA-based algorithm improved by 66.7%, 58.3%, 13.1% compared with WRR, Pick-KX and PSO-CO, respectively. Accordingly, applying the FWA-based load balancing algorithm in SDC-FN can reduce the task processing latency efficiently of the real-time mobile face recognition.

4.3 Influence of the Uplink Bandwidth for the Latency of SDC-FN

Since the bandwidth has a great impact on the transmission latency and can affect the total task processing time, we are motivated to investigate the influence of the band-width on the latency performance in SDC-FN in this section. We vary the uplink bandwidth of the cloud from 2 to 30 Mbps to evaluate how it affects the latency in SDC-FN by comparing it with the cloud-based architecture, and the task quantity is fixed to 20 Gb.The result is shown in Fig. 6. The figure illustrates that, the latency of cloud-based architecture decreases dramatically with the increase of the up-link bandwidth, but it stays steady when the up-link bandwidth is higher than 10 Mbps. While there are relatively little latency reduction in SDC-FN. Since in SDC-FN, a small amount of facial feature information after preprocessing is transmitted to the cloud, thus, the transmission latency would not decrease significantly when the uplink bandwidth of the cloud increases. And the latency in SDC-FN always keeps a low level, which indicates SDC-FN is more appropriate for real-time face recognition.

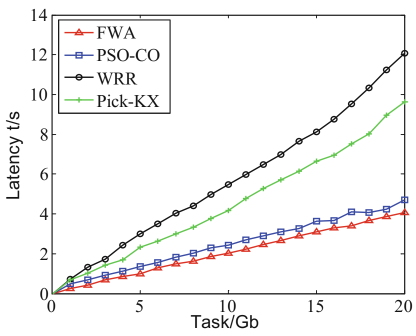

Fig. 5. Latency performance comparison among multiple load balancing algorithms in SDC-FN

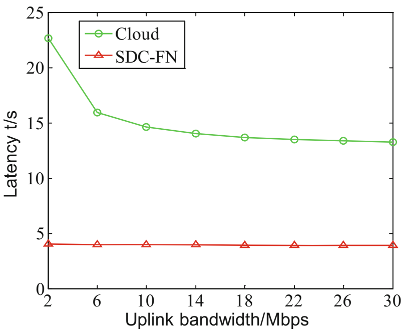

Fig. 6. Influence of the up-link bandwidth for the latency of SDC-FN

5 Conclusions

In this paper, we have introduced a novel network architecture which integrates fog computing and SDN to the cloud-based architecture in the real-time mobile face recognition to decrease the latency of the recognition service. Then, we set up a theoretical model of the load balancing problem in the software defined cloud-fog network (SDC-FN). On this basis, we proposed a SDN-based FWA centralized load balancing algorithm to balance the load for reducing latency efficiently. Simulation results reveal that: (1) The algorithm has good performance in reducing latency. (2) Applying the SDC-FN architecture in the real-time mobile face recognition scenario can meet the users' requirement of fast response time and improve the QoS. Our focus of the next step is to improve the performance of the FWA-based load balancing algorithm in the SDC-FN and implement the algorithm in the real SDC-FN platform.

Acknowledgments. This work was supported in part by National Natural Science Foundation of China (No. 61401331, No. 61401328), 111 Project in Xidian University of China (B08038), Hong Kong, Macao and Taiwan Science and Technology Cooperation Special Project (2014DFT10320, 2015DFT10160), The National Science and Technology Major Project of the Ministry of Science and Technology of China (2015zx03002006-003) and Fundamental Research Funds for the Central Universities (20101155739).

References

1. Truong, N.B., Lee, G.M., Ghamri-Doudane, Y.: Software defined networking-based vehicular adhoc network with fog computing. In: IEEE International Symposium on Integrated Network Management, Ottawa, pp. 1202–1207. IEEE Press (2015)
2. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC 2012, New York, pp. 13–16 (2012)

3. Aslam, S., Shah, M.A.: Load balancing algorithms in cloud computing: a survey of modern techniques. In: National Software Engineering Conference, pp. 30–35 (2015)
4. Panwar, R., Mallick, B.: Load balancing in cloud computing using dynamic load management algorithm. In: 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), pp. 773–778. IEEE Computer Society (2015)
5. Yi, S., Hao, Z., Qin, Z., Li, Q.: Fog computing: platform and applications. In: Third IEEE Workshop on Hot Topics in Web Systems and Technologies, pp. 73–78. IEEE Computer Society (2015)
6. Aazam, M., St-Hilaire, M., Lung, C.H., Lambadaris, I.: PRE-fog: IoT trace based probabilistic resource estimation at fog. In: 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 12–17 (2016)
7. Al Faruque, M., Vatanparvar, K.: Energy management-as-a-service over fog computing platform. IEEE Internet Things J. **3**, 161–169 (2016)
8. Tan, Y., Zhu, Y.: Fireworks algorithm for optimization. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) ICSI 2010. LNCS, vol. 6145, pp. 355–364. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13495-1_44
9. Bacanin, N., Tuba, M.: Fireworks algorithm applied to constrained portfolio optimization problem. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 1242–1249 (2015)
10. Imran, A.M., Kowsalya, M.: A new power system reconfiguration scheme for power loss minimization and voltage profile enhancement using fireworks algorithm. Int. J. Electr. Power Energy Syst. **62**, 312–322 (2014)
11. Hassan, M.A., Xiao, M., Wei, Q., Chen, S.: Help your mobile applications with fog computing. In: 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking-Workshops (SECON Workshops), pp. 1–6 (2015)
12. Li, X.Y., Tian, P., Kong, M.: A new particle swarm optimization for solving constrained optimization problems (in Chinese). J. Syst. Manag. **16**, 120–129 (2007)
13. Radojevi, B., Žagar, M.: Analysis of issues with load balancing algorithms in hosted (cloud) environments. In: 2011 Proceedings of the 34th International Convention, pp. 416–420 (2011)
14. Zhang, H., Liao, J.X., Zhu, X.M.: Advanced dynamic feedback and random dispatch load-balance algorithm (in Chinese). Comput. Eng. **33**, 97–99 (2007)