Service Scheduling Based on Edge Computing for Power Distribution IoT

Zhu Liu[1, 2, *], Xuesong Qiu[1], Shuai Zhang[2], Siyang Deng[2] and Guangyi Liu[3, *]

Abstract: With the growing amounts of multi-micro grids, electric vehicles, smart home, smart cities connected to the Power Distribution Internet of Things (PD-IoT) system, greater computing resource and communication bandwidth are required for power distribution. It probably leads to extreme service delay and data congestion when a large number of data and business occur in emergence. This paper presents a service scheduling method based on edge computing to balance the business load of PD-IoT. The architecture, components and functional requirements of the PD-IoT with edge computing platform are proposed. Then, the structure of the service scheduling system is presented. Further, a novel load balancing strategy and ant colony algorithm are investigated in the service scheduling method. The validity of the method is evaluated by simulation tests. Results indicate that the mean load balancing ratio is reduced by 99.16% and the optimized offloading links can be acquired within 1.8 iterations. Computing load of the nodes in edge computing platform can be effectively balanced through the service scheduling.

Keywords: PD-IoT, edge computing, service scheduling, load balancing strategy, ant colony algorithm.

1 Introduction

PD-IoT is a new type of power distribution network based on the integration of traditional power industry technology and the new generation information technologies, such as Internet of Things (IoT), cloud, big data analysis and artificial intelligence, Lü et al. [Lü, Luan, Liu et al. (2018)]. By incorporating the powerful sensing capability, various communication methods, big data analysis, and machine learning, PD-IoT provides a platform with built-in panoramic awareness of the distribution network, efficient and flexible data management tools, and agile software-defined applications development. With the vast access of new types of energy and loads, such as the distributed energy resource, multi-micro grids, electric vehicle charging stations, smart home, etc., the distribution network has become a bidirectional energy node [Haider, See and Elmenreich (2016); Chakraborty, Iu and Lu (2015); Yu, Zhong, Xie et al. (2016)]. As

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China.

[2] State Grid Information & Telecommunication Group Co., Ltd., Beijing, 102211, China.

[3] Global Energy Interconnection Research Institute North America, San Jose, CA, 95134, USA.

[*] Corresponding Authors: Zhu Liu. Email: liuzhu_558@139.com;
Guangyi Liu. Email: guangyi.liu@geirina.net.

a result, quantity of data and business for power distribution skyrocket, and the power distribution is urgently needing a higher service quality, lower time-delay and stronger interactivity [Primadianto and Lu (2017); Dehghanpour, Wang, Wang et al. (2019)]. The centralized cloud computing model is faced with the challenge of higher computing, storage and bandwidth demands [Yaghmaee, Leon-Garcia and Moghaddassian (2018); Bera, Misra and Rodrigues (2015)].

In order to alleviate the processing pressure and eliminate the bottleneck of computing and communication for the centralized cloud model, edge computing is introduced [Li, Ota and Dong (2018); Satyanarayanan (2017)]. Edge computing is a distributed computing paradigm, which manages the massive grid data from grid devices and sends the key results to the cloud. It expends the scope and ability of data collection and management for the cloud. Meanwhile, since the edge computing is displaced near the data source, the load of communication between grid device and cloud is reduced, especially for the large-scale data, such as the video data, warning message, etc. [Liu, Yang, Jiang et al. (2019); Shi, Cao, Zhang et al. (2016)].

Due to the limited physical computation ability of single computing node, the computing and storage capacity may not achieve the demand of real-time processing faced to some instantaneous grid failures or abnormal fluctuation. It is a potential safety hazard for power distribution. In this case, good network structure and efficient offload strategy are essential [Ruan, Liu, Qiu et al. (2018)]. Previous studies about service scheduling is mainly focused on a certain problem, without considering the diversity of multiple tasks and loads of the edge computing platform in smart grid network. For example, min-min algorithm is used to assign the service to a node with the shortest execution time [Moggridge, Helian, Sun et al. (2017)]. It pursues to finish the tasks as soon as possible and does not consider the load situation of the receiving node, which may cause the unbalance and reduce the resource utilization. Martino et al. [Martino and Mililotti (2004)] proposes a service scheduling method with genetic algorithm, the main idea is using First Come First Service rule to rank the order of tasks in the queue. However, this rule has the disadvantage of poor fairness. Round Robin method is another classic algorithm with polling the nodes [Ghomi, Rahmani and Qader (2017)], however it may affect the performance of CPU due to the frequent changes.

This paper presents a service scheduling method based on edge computing for the PD-IoT. Architecture of the PD-IoT and structure of the service scheduling system are designed. A novel load balancing strategy and ant colony algorithm are embedded in the offload method. The validity is simulated.

2 Architecture design

2.1 Architecture design of PD-IoT

Architecture of the PD-IoT with edge computing platform is depicted in Fig. 1, which is a four-tier cloud-network-edge-terminal model with global security and communication protocol system.

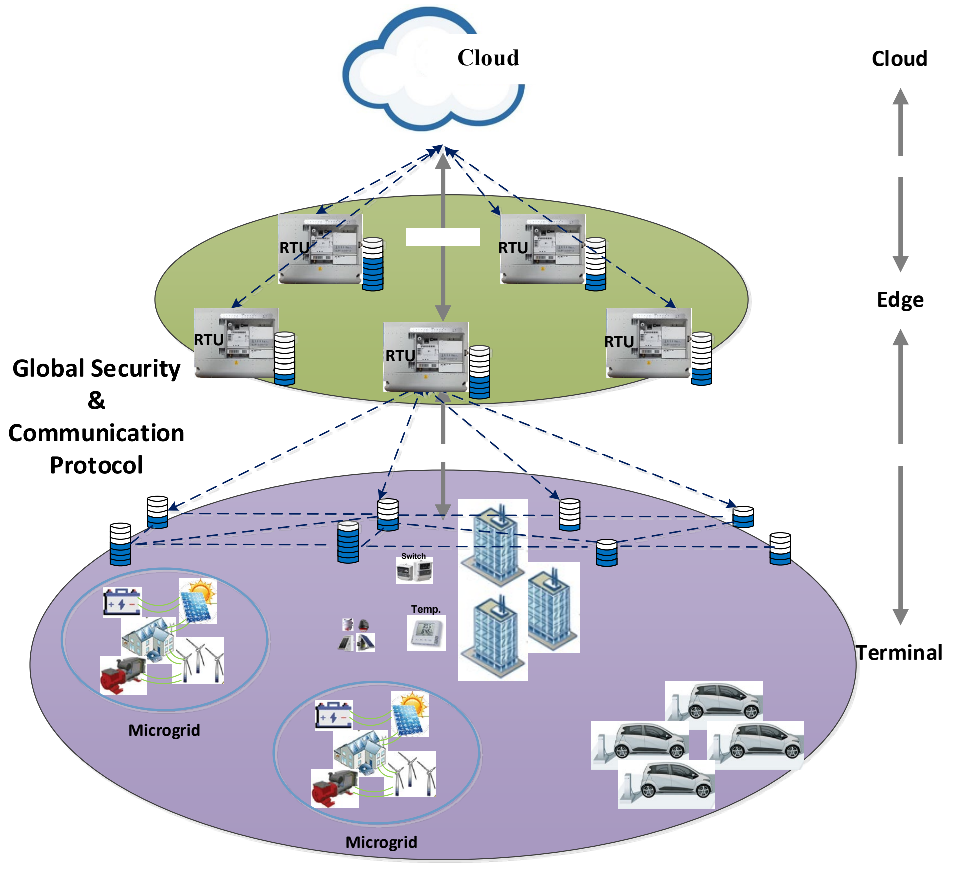

Figure 1: Architecture of PD-IoT with edge computing platform

Terminal: The "terminal" provides the awareness of a PD-IoT system. It is responsible for providing the operating status, electrical status, environmental status, and other auxiliary information of the distribution network to the "edge" or the "cloud". It is also responsible for executing grid commands and controls. The PD-IoT terminals include multi-micro grids, charging piles, residential or industrial equipment, various electrical sensors and environmental sensing units, et al. Part of the terminals have some computation capacity. PD-IoT terminals should support the standardization of the services, plug-and-play, interconnections between devices, and security.

Cloud: The "cloud" is the master platform, which adopts technologies such as cloud computing, big data analysis, and machine learning. Such a cloud platform provides a full migration path for various distribution management services using the micro-services architecture. The PD-IoT cloud includes the infrastructure as a service (IaaS) layer, the platform as a service (PaaS) layer, and the software as a service (SaaS) layer [Kavis (2014)], which aims to serve as a platform solution for PD-IoT systems. Meanwhile, the PD-IoT cloud should decouple software applications from the underlying hardware, applications from data, and serve to meet massive terminal equipment connections, dynamic resource allocation, and future business needs.

Network: The "network" provides data transmission among the cloud, the edge and the terminal for the PD-IoT. It is a communication network for consolidating and exchange massive grid data and business. The PD-IoT network includes a telecommunication network and a local communication network. Components of the network, global security and communication protocol system are shown in Fig. 2. The telecommunication network provides communication channels between the cloud and edge nodes. The communication methods include EPON, industrial Ethernet, electric wireless private networks, wireless public network, and the satellite communication, etc. The local communication network provides communication channels between the edge nodes and massive terminals. The communication methods include power line communication (PLC), low power wireless, Ethernet, and serial communication, etc. The PD-IoT network aims to be fully adaptable, widely interconnected and with efficient bandwidth.

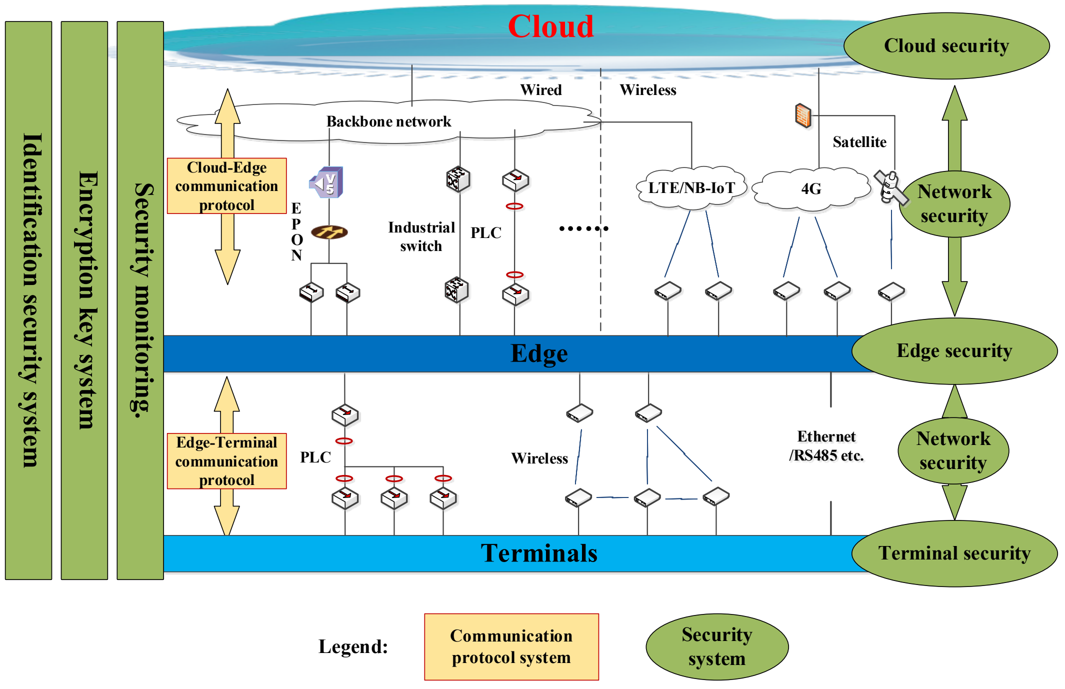

Figure 2: Components of the network, global security and communication protocol system

Security: The security system mainly includes cloud security, network security, edge security, terminal security, which adopts security measures such as the unified identification security system, unified encryption key system and provides unified security monitoring. The PD-IoT security system should be lightweight.

Communication protocol: The communication protocol includes the cloud-edge communication protocol and the edge-terminal communication protocol. Due to resources and communication bandwidth restraints and the business needs to accommodate for massive IoT connections, the PD-IoT communication protocol among the cloud-edge-terminal requires simple and efficient data coding methods, especially for the edge-terminal communication protocol.

Edge: The "edge" is a distributed intelligent agent close to the data source. It is an extension of the PD-IoT cloud. Remote terminal unit is a typical edge device in power distribution. In this paper, the edge means the edge computing platform, which is composed of the edge device and all of the intelligent PD-IoT terminals which have computation capacity (named as computing nodes). The edge computing platform virtualizes the computing nodes to form computation, data storage, and network resource pools with flexible computing, storage scalability, dynamic load balancing ability. The PD-IoT edge should support agile software defined App development and possible fast deployment to support evolving distribution applications.

2.2 Structure of edge computing platform design

The structure of the service scheduling system is shown in Fig. 3. The computing, storage of the edge and terminals are virtualized as computing resource pool and storage resource pool, which are distributed and scheduled according to the task management (includes task monitoring, task scheduling and task computing). In normal state, the edge and terminals work in coordination according to the established allocation. Once the fault state occurs, the amount of information and calculation will increase suddenly. If the computing load of a node surpass the overload line, the node will send a requirement to the edge computing platform for a service scheduling collaboration. The platform selects the optimal service receiving computing node and the shortest transfer link according to the load balancing strategy in order to reduce the time-delay and improve the efficiency of edge computing platform.

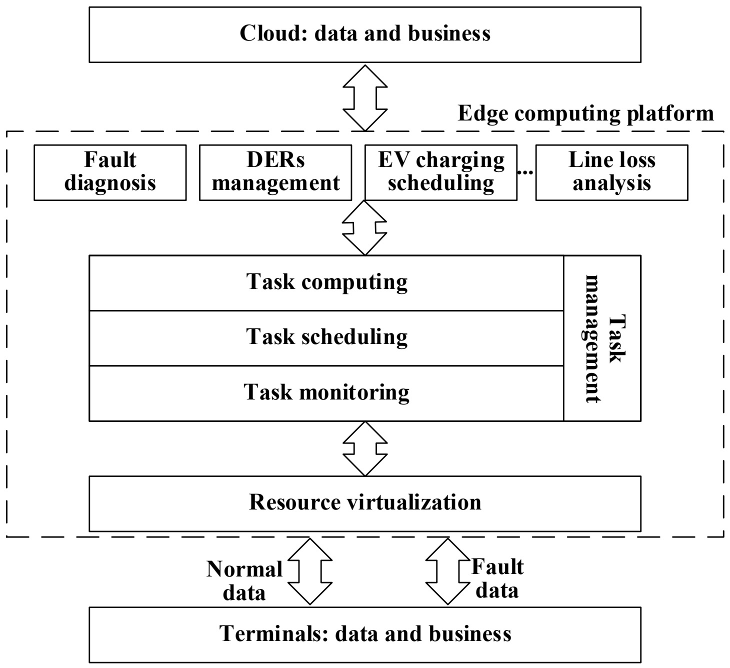

Figure 3: Structure of service scheduling system for edge computing platform

3 Algorithm design

3.1 Flow chart of service scheduling

The flow chart of the service scheduling method is shown in Fig. 4. Once a service is issued from a computing node, the node determines that, whether its own load exceeds the high threshold after receiving the task. If not, the service could be accepted and the state of load is updated. Otherwise, the system should select the optimal low-load node that could receive the request based on the load balancing strategy and the shortest transfer link based on ant colony algorithm. The chosen one accepts the service and update its load state. This service scheduling can be used to avoid the overload of computing nodes and provide real-time response to the service requisitions.

3.2 Receiver selection: load balancing strategy design

A strategy of load balancing for multi-nodes and loads in PD-IoT is designed here. The calculation resources and load state of each computing node in the edge computing platform are quantified by percentage (from 0% to 100%) according to its computation capacity. Each computing node has its own thresholds of full-load, high-load and normal-load (marked as F-threshold, H-threshold, and N-threshold), and has three load states: low-load state (lower than N-threshold), normal-load state (between N-threshold and H-threshold), and high-load state (higher than H-threshold). According the Locality of Reference [Denning (2006)], the nodes with higher load states would continue to receive a large amount of information and service in the following period of time. In view of this, the load balancing strategy should offload the service from the nodes with high-load states as much as possible. The purpose is that the original nodes with high-load states enable to become the nodes with low-load states, while other nodes stays out of the high-load state. The detailed strategy is listed as follows:

1) Select the nodes with high-load states (recorded as node group A) and sort them in descending order of load. The strategy would transfer the heaviest load of the rest. The total loads which surpass the H-threshold of the node group A is marked as X1 (the first grade offloading service), and the total loads between N-threshold and H-threshold of node group A is noted as X2 (the second grade offloading service);
2) Select the nodes with low-load states (recorded as node group B) and arrange them in ascending order according to the load. The strategy would choose the lightest node to receive the service at first. The total margin load for H-threshold of node group B is Y1 (the first grade margin load) and the total margin load between the H-threshold and F-threshold is Y3 (the third grade margin load);
3) Select the nodes with normal states (recorded as node group C) and sort them in ascending order according to the load. The strategy would choose the lightest node to receive the service at first. The total margin load for H-threshold of node group C is Y2 (the second grade margin load);
4) Determine the magnitude of X1, X2 and Y1, Y2, Y3, then chose the transition state.

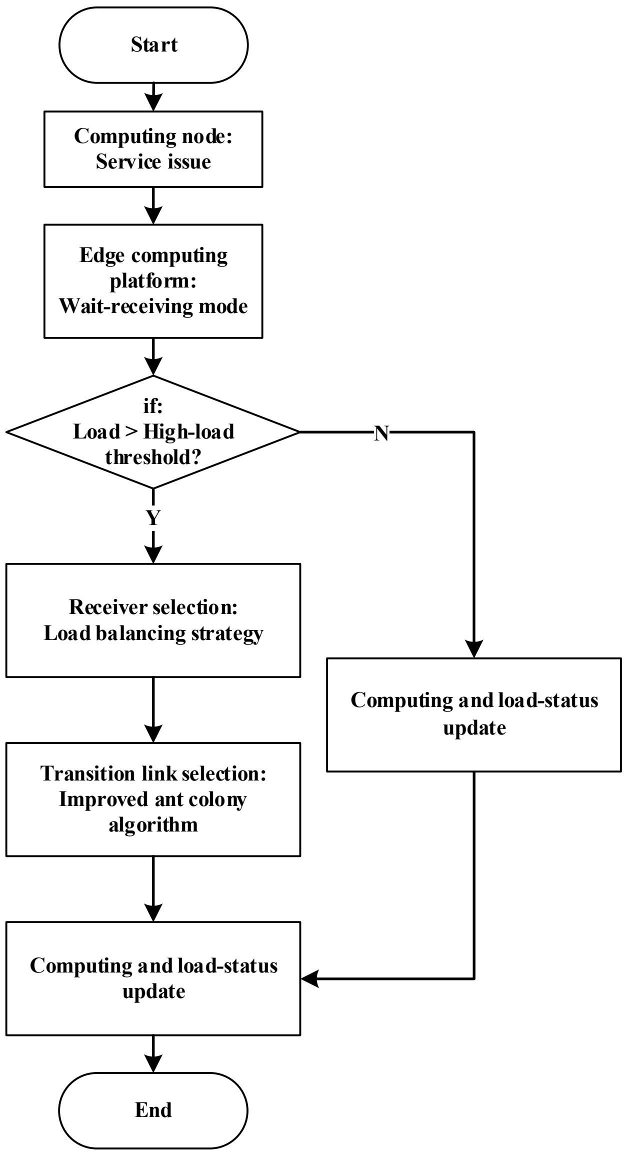
Start
Computing node: Service issue
Edge computing platform: Wait-receiving mode
if: Load > High-load threshold?
N
Y
Receiver selection: Load balancing strategy
Transition link selection: Improved ant colony algorithm
Computing and load-status update
Computing and load-status update
End

Figure 4: Flow chart of service scheduling

5) State 1 (Y1 as receivers): X1+X2<Y1. Loads X1 and X2 in group A are transferred to Y1 in node group B. So, all nodes in group A are with low-load states, nodes in group B are out of the high-load states, and the states of nodes in group C stay the initial state. No high-load exists.
6) State 2 (Y1 and Y2 as receivers): X1+X2>Y1 while X1+X2<Y1+Y2. Loads X1 and X2, are transferred to the Y1 in group B and Y2 in group C. So, the nodes in group A are with low-load state, nodes in groups B and C are out of the high-load states. No high-load exists;
7) State 3 (Y1, Y2 and Y3 as receivers, Yellow warning): X1+X2>Y1+Y2 while

X1+X2<Y1+Y2+Y3. Loads X1 and X2 in group A are transferred to Y1, Y2 and Y3 in groups B and C. So, the nodes in group A are with low-load state, nodes in group C are out of the high-load states. Part of the nodes in group B are with high-load states. Yellow warning: reduce the amount of grid data from terminals and prepare to request cloud-edge cooperation;

8) State 4: (Y1, Y2 and Y3 as receivers, Orange warning):X1+X2>Y1+Y2+Y3 while X1<Y1+Y2+Y3. All service of X1 and parts of X2 in group A are transferred to Y1, Y2 and Y3 in groups B and C. Then, the nodes in groups A and C are out of high-load states, all nodes in group B are with high-load states. Orange warning: reduce the amount of grid data from terminals and start cloud-edge cooperation;
9) State 5: (Y1, Y2 and Y3 as receivers, Red warning): X1>Y1+Y2+Y3. Parts of loads X1 are transferred to Y1, Y2 and Y3 in groups B and C. Then, part nodes in group A are still with high-load states, all nodes in group B are with high-load states, and nodes in group C are out of high-load states. Red warning: reduce the amount of grid data from terminals and start cloud-edge cooperation. If the state of red warning is continuous, the computing nodes in edge computing platform needs to be upgraded.

To quantify the success rate of the load balancing, a parameter called load balancing ratio H is introduced here:

$$H = \frac{Reduced\ number\ of\ loads\ with\ high\text{-}load\ states\ after\ load\ balancing}{Number\ of\ loads\ with\ high\text{-}load\ states\ before\ load\ balancing} \times 100\% \quad (1)$$

3.3 Transmission link selection: ant colony algorithm design

When the offloading service and the receiver node are selected, the optimal transmission link should be chosen. The transmission links between two computing nodes in edge networks are multiple. The transfer speed, attenuation rate, noise resistance and transmission success rate are influenced by the media of channel, bandwidth, physical distance and the number of communication nodes. If these mentioned elements are taken as the weight of the link, this problem could be simplified to a shortest path optimization problem with weighted undirected graph.

Traditional path planning algorithms, such as Dijkstra algorithm and A* algorithm, are simple to operate, but not suitable for large-scale and complex real-world problems. Moreover, they cannot meet the requirements of time hysteresis. In view of this problem, ant colony algorithm (ACA) is selected to choose the optimal transmission links. ACA is a heuristic algorithm for simulating the foraging behavior of real ant colonies. The key point of this behavior is the indirect communication between ants by means of chemical pheromone trails [Blum (2005)]. At the beginning of searching for foods, ants explore the paths in a random manner. Then they leave a chemical pheromone trail on the ground while moving. Other ants can smell it and tend to choose the paths marked by strong pheromone concentrations. The shorter the path is, the more ants can return in unit time. Thus more pheromone can be left. The dense pheromone will attract even more ants as a positive feedback. After iterations, every ant will choose the shortest path. The corresponding algorithm flow chart is shown in Fig. 5.

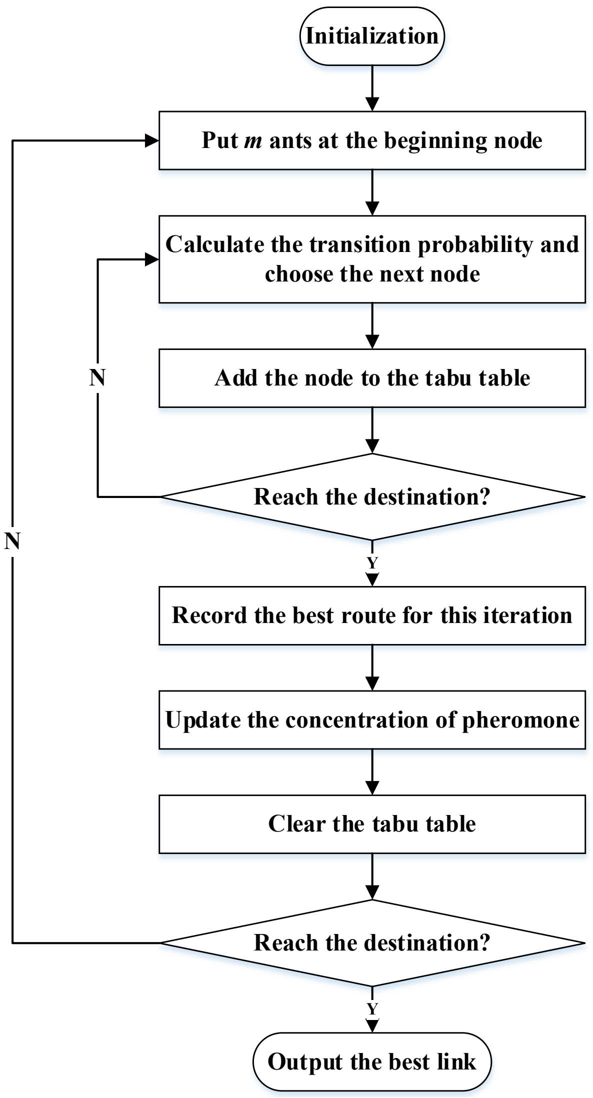
Initialization
Put *m* ants at the beginning node
Calculate the transition probability and choose the next node
Add the node to the tabu table
N
Reach the destination?
N
Y
Record the best route for this iteration
Update the concentration of pheromone
Clear the tabu table
Reach the destination?
Y
Output the best link

Figure 5: The flow chart of ant colony algorithm

4 Simulation results and analysis

4.1 Receiver selection simulation

Suppose that the service scheduling system has a total of 50 computing nodes. The load states of these nodes were with random values between 5%-85%. The F-threshold, H-threshold, and N-threshold of each computing node were set as random numbers between 95%-98%, 65-75% and 30%-40%, respectively. Based on the novel load balancing strategy, simulation result of the load balancing is presented in Fig. 6.

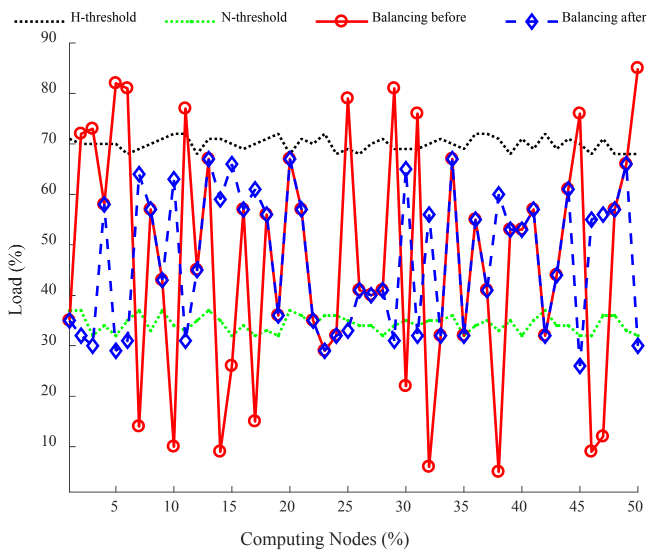

Figure 6: Simulation of single test for load balancing

As can be seen from Fig. 6, a total of 10 computing nodes are with high-load states before load balancing. After the load balancing, all of the computing nodes in this edge computing platform are in normal-load or low-load states. The loads of nodes with high-load are efficiently transferred, the load balancing ratio is 100%.

In order to avoid the coincidence of a single test, 100 times of repeated experiments utilizing the load balancing strategy were simulated. The F-threshold, H-threshold, and N-threshold of these 50 computing nodes were invariable in the 100 tests, which were identical to the values in the first single test in Fig. 6. To increase the initial loads of these nodes, the initial loads were set as the random values between 20%-90% in these repetitive tests. The simulation results are shown in Fig. 7. During these 100 times experiments, the nodes with high-load states were still existed occurred in 4 times tests using the load balancing. We can find that these cases were happened in the conditions that the initial numbers of nodes with high-load states, before load balancing, were relatively higher. The mean number of the nodes with high load is 14.23 before the load balancing, but it can be reduced to 0.12 using the load balancing strategy. The mean load balancing ratio is 99.16%.

4.2 Transmission link selection simulation

Suppose that the links from the computing node to be unloaded (denoted as node A) and the receiving node (denoted as node O) involves 15 nodes (denoted as node A-O). Connections between the nodes represents the transmission links (25 links in total), and the numbers on the lines show the weights. The smaller the weight is, the better the link is. In this simulation, the weights were generated by random sampling from 1 to 10. The main experimental parameters of ant colony algorithm are listed in Tab. 1.

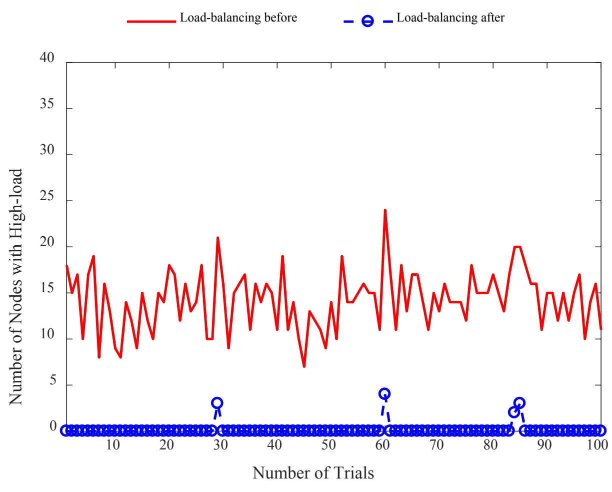

Figure 7: Simulation of 100 tests for load balancing

Table 1: Parameters of ACA

Parameters	Values
m , the number of ants	50
α , importance of pheromone	1
β , importance of heuristic information	5
ρ , evaporation rate	0.1
$Iter_{\max}$, max number of iteration	50

Since the pheromone concentrations of each link are equal at the beginning, the first selection of route in the first iteration is only related to the probability of the path which is inversely proportional to the weights. In order to avoid falling into local optimum, it is necessary to ensure that there are enough ants choosing different paths initially. The topology of links between the nodes A and O and the simulation results using the ACA is shown in the Fig. 8. The highlighted transmission link, chosen by ACA, involves 7 nodes (nodes A-B-C-D-E-F-O) and the total weights of this link is 23. It is the best transmission link from node A to node O. The results prove that ACA is an effective method in link selection.

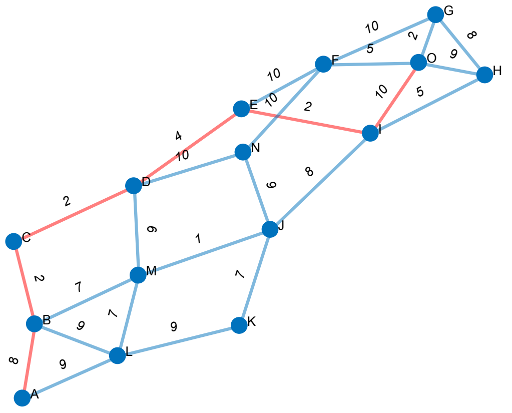

Figure 8: Simulation of single test for transmission link selection

50 times of repetitive tests were performed by changing the weights of each link. The number of iterations which finds the optimum were presented in Fig. 9. The abscissa of Fig. 9 indicates the number of trials and the ordinate represents the number of iterations when the optimal transmission link was found. It can be seen that the optima can be found within 5 iterations in most of the time, and the average iteration number is 1.8.

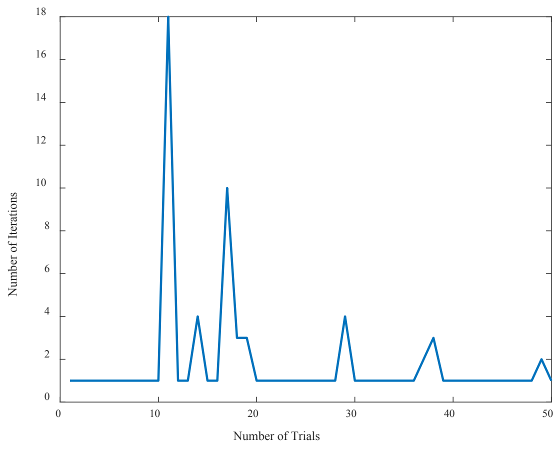

Figure 9: Simulation of 50 tests for best transmission link selection

5 Conclusions

A service scheduling method based edge computing platform for PD-IoT is proposed in this paper. The architecture of the PD-IoT with edge computing platform and the structure of the service scheduling system are designed. A novel load balancing strategy and ant colony algorithm are utilized in this service scheduling method. Simulation studies of this method have been conducted. Results demonstrate that the mean load balancing ratio can be reduced by 99.16% and optical transmission links can be found within 1.8 iterations. This means that the service scheduling method designed is efficient.

Acknowledgement: This work was supported by the National Natural Science Foundatio n of China (Grant: 61702048).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Bera, S.; Misra, S.; Rodrigues, J. J. (2015): Cloud computing applications for smart grid: a survey. *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1477-1494.

Blum, C. (2005): Ant colony optimization: introduction and recent trends. *Physics of Life Reviews*, vol. 2, no. 4, pp. 353-373.

Chakraborty, C.; Iu, H. H. C.; Lu, D. D. C. (2015): Power converters, control, and energy management for distributed generation. *IEEE Transactions on Industrial Electronics*, vol. 62, no. 7, pp. 4466-4470.

Dehghanpour, K.; Wang, Z.; Wang, J.; Yuan, Y.; Bu, F. (2019): A survey on state estimation techniques and challenges in smart distribution systems. *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 2312-2322.

Denning, P. J. (2006): The locality principle. *Communication Networks and Computer Systems: A Tribute to Professor Erol Gelenbe*, pp. 43-67.

Di Martino, V.; Mililotti, M. (2004): Sub optimal scheduling in a grid using genetic algorithms. *Parallel Computing*, vol. 5, no. 30, pp. 553-565.

Ghomi, E. J.; Rahmani, A. M.; Qader, N. N. (2017): Load-balancing algorithms in cloud computing: a survey. *Journal of Network and Computer Applications*, vol. 88, pp. 50-71.

Haider, H. T.; See, O. H.; Elmenreich, W. (2016): A review of residential demand response of smart grid. *Renewable and Sustainable Energy Reviews*, vol. 59, pp. 166-178.

Kavis, M. J. (2014): *Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS)*. John Wiley & Sons.

Li, H.; Ota, K.; Dong, M. (2018): Learning IoT in edge: deep learning for the internet of things with edge computing. *IEEE Network*, vol. 32, no. 1, pp. 96-101.

Liu, Y.; Yang, C.; Jiang, L.; Xie, S.; Zhang, Y. (2019): Intelligent edge computing for iot-based energy management in smart cities. *IEEE Network*, vol. 33, no. 2, pp. 111-117.

Lü, J.; Luan, W.; Liu, R.; Wang, P.; Lin, J. (2018): Architecture of distribution internet of things based on widespread sensing & software defined technology. *Power System Technology*, vol. 42, no. 10, pp. 3108-3115.

Moggridge, P.; Helian, N.; Sun, Y.; Lilley, M.; Veneziano, V. et al. (2017): Revising max-min for scheduling in a cloud computing context. *IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 125-130.

Primadianto, A.; Lu, C. N. (2017): A review on distribution system state estimation. *IEEE Transactions on Power Systems*, vol. 32, no. 5, pp. 3875-3883.

Ruan, L.; Liu, Z.; Qiu, X.; Wang, Z.; Guo, S. et al. (2018): Resource allocation and distributed uplink offloading mechanism in fog environment. *Journal of Communications and Networks*, vol. 20, no. 3, pp. 247-256.

Satyanarayanan, M. (2017): The emergence of edge computing. *Computer*, vol. 50, no. 1, pp. 30-39.

Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. (2016): Edge computing: vision and challenges. *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646.

Yaghmaee, M. H.; Leon-Garcia, A.; Moghaddassian, M. (2018): On the performance of distributed and cloud-based demand response in smart grid. *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5403-5417.

Yu, R.; Zhong, W.; Xie, S.; Yuen, C.; Gjessing, S. et al. (2016): Balancing power demand through EV mobility in vehicle-to-grid mobile energy networks. *IEEE Transactions on Industrial Informatics*, vol. 12, no. 1, pp. 79-90.