Load Balancing in Software-Defined Networks Based on Particle Swarm Optimization

Haiyan Zhang(✉), Liren Zou, and Yilong Xie

School of Computer Technology, Beijing Institute of Technology (Zhuhai), Zhuhai 519085, China
2552974138@qq.com

Abstract. Nowadays, as Software-Defined Networking (SDN) gains prominence, Load Balancing (LB) for SDN assumes significant importance. By allocating network traffic among resources efficiently, LB ensures that no individual resource is burdened, thereby optimizing the overall performance. Based on the analysis of SDN Flow network forwarding mechanism, this paper applies Particle Swarm Optimization (PSO) algorithm to the data center's traffic scheduling based on load balancing. First, the SDN load balancing problem is abstracted as an integer Linear programming model, which maximizes the average link bandwidth utilization on the basis of ensuring network delay. PSO algorithm is applied to optimize the load balancing problem, and the optimization algorithm is run in the SDN controller. The simulation experiment by Mininet shows that the SDN load balancing algorithm based on Particle swarm optimization can effectively balance the network load and improve the network performance.

Keywords: SDN · Load Balancing · Particle Swarm Optimization

1 Introduction

Over the past few years, the development of many new technologies, such as the Internet of Things and cloud systems, has experienced rapid growth. Consequently, it becomes imperative to augment the traditional Internet Protocol network to effectively manage the substantial volume of network traffic.

Software-Defined Networking (SDN) [1] has transformed the traditional approach to network management by dismantling vertical integration of network components, divorcing network logic control from underlying switches and routers, promoting centralization of network control, and incorporating programmability of network operations.

However, SDN has a centralized control architecture, which raises concerns about reliability, fault tolerance, scalability and interoperability [2]. Load Balancing ranks among the most crucial tasks for enhancing network performance, robustness and scalability. Therefore, the combination of multi-objective optimization is essential for identifying an optimization point from a set of non-dominated points. SDN load balancing based on metaheuristic has become a hot research topic [3].

V. C. M. Leung et al. (Eds.): Qshine 2023, LNICST 573, pp. 472–480, 2024.
https://doi.org/10.1007/978-3-031-65126-7_39

The Particle Swarm Optimization (PSO) algorithm has been recognized to be effective for its efficiency in SDN load balancing, which to be compared to several other existing optimization techniques [4].

2 Related Work

Load balancing in Software-Defined Networking (SDN) has become an prominent research topic in recent times. In the experiment of [5], the authors provided a classification method for load balancing in SDN, delving into various objectives, such as response time optimization, overall resource optimization throughout, and identification of bottle necks. Their taxonomy distinguished between load balancing in the control plane and load balancing in the data plane. [6] proposed an approach of multiple SDN controllers for dynamic load balancing. And K. Sridevi [7] uses Artificial Bee Colony for distributed controller load balancing.

There are also many studies on load balancing for traffic. The authors of [8, 9] use conventional approaches such as the OSPF and EMCP algorithm, to optimize the network. But somehow, this static hashing method may cause multiple large streams to be divided into the same path, leading to collisions and causing network congestion. Due to a lack of historical experience, numerous studies have utilized supervised learning methods in SDN for achieving intelligent network management [10, 11]. However, there are significant challenges for supervised learning, because of the reliance on abundant training datasets and the sluggish decision-making process in dynamic network scenarios. Heuristic methods have unique advantages in traffic load balancing, such as a study by [12, 13], that proposed a dynamic load balancing algorithm based on genetic ant colony optimization, and a study by [14] that proposed genetic optimization for traffic loading using SDN. In [15], a modified PSO algorithm was proposed to integrate SDN with fog computing, in order to solve the load balancing issue. Two multi-objective particle swarm optimization methodologies are proposed in [4], which are distance Angle Multi-Objective PSO and Angle Multi-Objective Particle Swarm Optimization.

3 Load Balancing in SDN

The architecture of SDN has shifted the control decision process on the controller, whereas the switch relies entirely on the flow rules injected by the controller to forward data to the device [16]. Transfer the load to a specific switch, and the controller makes decisions based on this.

SDN based load balancing technology has three major advantages:

1) The controller in the SDN architecture can collect global network topology, link available bandwidth, and other resources for learning through the OpenFlow protocol, and develop a better load balancing strategy from a global perspective based on the bandwidth requirements of the data flow.
2) Centralized logical control can abstract all OpenFlow forwarding devices into a whole and be uniformly distributed by the controller to forwarding devices such as switches, without the need for complex configurations of devices with different standards in traditional networks.

3) The business orchestration technology in SDN architecture can use software to develop load balancing strategies.

4 Method

4.1 Optimization Objective Functions

The essence of the traffic load balancing problem is to select an appropriate path according to the path information to schedule the flow from the source node to the target node. Here, the flow path problem can be abstracted into integer linear programming model for solution. The specific model is as follows:

For the classic Fat-Tree data center network topology, it can be abstracted as a directed graph, where H and S distributions represent the set of host nodes and switch nodes, while E denotes the collection of links. The network contains m links, where the links can be expressed l_{ij}, i, j $\in$ H $\cup$ S. Additionally, define the maximum load of the link as C_{ij}. There are n traffic in the network, and the set of traffic can be represented as $F = \{f_1, f_2, \cdots, f_n\}$, One traffic f_k is defined as a triplet $\left(s^k, d^k, b^k\right)$, where s^k represents the source host node, d^k represents the destination host node, and b^k represents the bandwidth occupied by traffic f_k.

The optimization goal is to maximize the average link bandwidth utilization, which means maximizing the utilization of network data while avoiding network congestion. Therefore, the objective function is:

$$\max\left(\frac{\sum_{\{(i,j)\in E\}}\left(\sum_{1\le k\le n} b_{ij}^k / C_{ij}\right)}{m}\right) \tag{1}$$

Among them, $\sum_{1\le k\le n} b_{ij}^k$ represents the actual load of l_{ij}, b_{ij}^k represents the actual bandwidth of traffic f_k on link l_{ij}. The constraints are as follows:

$$\sum\nolimits_{1\le k\le n} b_{ij}^k \le C_{ij}, \forall(i,j) \in E \tag{2}$$

$$0 \le \forall b_{ij}^k \le C_{ij}, i, j \in H \cup S, k \in \{1, 2, \cdots, n\} \tag{3}$$

$$\sum\nolimits_{\{j:(s^k,j)\in E\}} b_{s^k j}^k - \sum\nolimits_{\{j:(j,s^k)\in E\}} b_{j s^k}^k = b^k, k = 1, 2, \cdots n \tag{4}$$

$$\sum\nolimits_{\{j:(d^k,j)\in E\}} b_{d^k j}^k - \sum\nolimits_{\{j:(j,d^k)\in E\}} b_{j d^k}^k = -b^k, k = 1, 2, \cdots n \tag{5}$$

$$\sum\nolimits_{\{j:(i,j)\in E, i\ne s^k, d^k\}} b_{ij}^k = \sum\nolimits_{\{j:(j,i)\in E, i\ne s^k, d^k\}} b_{ji}^k, k = 1, 2, \cdots n \tag{6}$$

Formula (2) represents the traffic capacity constraint, which means that the total flow on any link should not exceed the link capacity. Formula (3) represents the bandwidth constraint of the traffic, which means that the bandwidth occupied by the traffic f_k should be greater than or equal to 0, cannot be negative, and less than the link capacity. Formulas

(4)–(6) define a traffic conservation constraint, which means that the traffic of f_k from the source host to the destination host is equal to the traffic of any node in the path.

When forwarding traffic, map the model to the particle swarm optimization algorithm to obtain a specific scheduling plan. When the network traffic is at the peak, in order to avoid uneven link responsibility caused by network congestion, a load balance degree of the whole network is set to determine whether the current network is in the load balance state. This article cites the concept of standard deviation in mathematics and uses discrete programs to reflect equilibrium. At the same time, to avoid unnecessary rerouting caused by short-term extreme values, the mean over a period of time is used as the result. The specific definition of load balancing is as follows:

$$\delta(t) = \frac{1}{P}\sum_{t=T-P}^{T}\left(\sqrt{\frac{1}{m}\sum_{l=1}^{m}\left(\overline{load(t)} - load_l(t)\right)^2}\right) \tag{7}$$

Among them, P represents the statistical period, T represents the current time, $\overline{load(t)}$ represents the average load of all d-rated links at time t, and $load_l(t)$ represents the real-time load of link l at time t.

4.2 Particle Swarm Optimization

PSO is mainly used to solve optimization problems and is one of the meta heuristic algorithms based on natural heuristic proposed by Kennedy and Eberhart. In particle swarm optimization, candidate solutions (usually referred to as particles) are continuously refined to seek the optimal solution. This refinement is achieved by adjusting the motion of particles, which are influenced by their understanding of the most determined local position in the search space.

At the initial state, all swarm particles are randomly positioned in a multidimensional search space. The space's dimensions of which depends on the problem being addressed. The particles navigate through the space by adjusting their velocities and subsequently changing their positions. Each particle possesses its own current position and velocity attributes. The location of a particle represents a potential solution to the problem. To update their positions, particles utilize controlled rules to adjust their velocities, enabling them to move toward better locations.

Within PSO, every particle maintains two values: (1) its personal best position (localBest) and (2) the overall best position achieved by the entire group (globalBest). These values are utilized for velocity updates, as illustrated in Eqs. (8) and (9) [17, 18]. Here, C_1 and C_2 are positive numbers known as acceleration coefficients, while r_{1d} and r_{2d} denote two uniformly distributed numbers within the [0, 1]. P_{id} and P_{gd} prefer to the particle's localBest and globalBest positions, respectively. As the velocity is adjusted, the formula for calculating the position of particles is as follows

$$V_{id}(t+1) = V_{id}(t) + C_1 r_{1d}(P_{id} - X_{id}) + C_2 r_{2d}\left(P_{gd} - X_{id}\right) \tag{8}$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1) \tag{9}$$

5 Experimental Results and Analysis

5.1 Experimental Environment Configuration

Mininet, a network simulation tool developed by Stanford University, is a lightweight software-defined platform that supports various southbound protocols like OpenSwitch and OpenFlow. It allows you to build entire networks, consisting of switches, links, and hosts, on a single physical device.

This article describes the use of a Python-based Ryu controller to build an SDN network through Mininet. The experiment is conducted on a HP laptop running a VirtualBox virtual machine with Ubuntu 18.04 as the operating system. Scapy is utilized to inject traffic into the virtual network, simulating network traffic. The simulation parameters for this experimental environment are presented in Table 1.

Table 1. Parameters

Parameter	Description
OS	Ubuntu 18.04
CPU	Ryzen 5-3550
RAM	16G
Simulator Tool	Mininet 2.3.1
Mininet 2.3.1	RYU 4.30
Protocol	OpenFlow V1.3
Traffic generator	Scapy

To validate the performance of the PSO algorithm, the average split bandwidth and end-to-end delay were used as performance indicators. Experimental tests were conducted on the Fat-Tree network topology (Fig. 1), and the PSO algorithm was compared with ECMP and Round Robin.

As depicted in Fig. 1, comprises of 20 switches and 16 hosts interconnected in a random configuration. All these hosts and switches are located within the infrastructure layer, and the controllers are located within the control layer. Any source host communicates with the target host by connecting their switches. If the source host and the destination host in the flow table are not connected, send a packet to the controller_In message. The controller uses the PSO algorithm to determine the optimal and least loaded path, and injects flow entries along that path into the switch to forward user requests. The specific parameters used in the network simulation environment are outlined in Table 2.

Assuming the source host is H1 from pod 1, the destination host is H8 from pod 2, and the data packet is sent from the source to the destination. It is obvious that there are four ways to reach the destination.

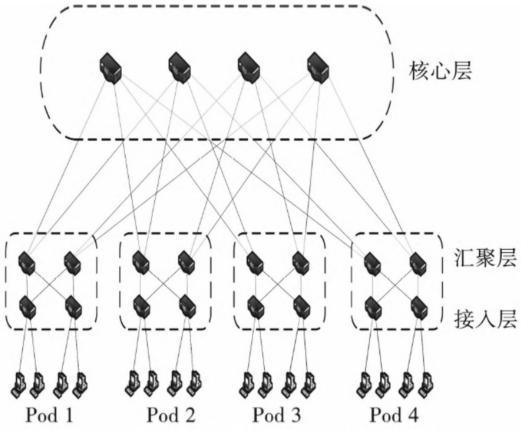

Fig. 1. Fat-Tree network topology

Table 2. Network environment parameters

Parameter	Description
Link Bandwidth	10 Mbps
Switch maximum buffer queue	1000
Traffic timeout	5 s

In the experiment, virtual hosts in the network communicate randomly with equal probability, and the parameter settings of the PSO algorithm directly affect its performance. The parameters of the algorithm have been determined through the experiment as shown in Table 3.

Table 3. PSO algorithm parameters

Parameter	Description
Number of particles	20
Number of iterations	100
w	0.5
c1	1/3
c2	2/3

5.2 Results and Analysis

In this study, we compared it with two commonly used load balancing algorithms: loop algorithm and ECMP algorithm. Extract the results of each load balancing algorithm in the following order:

1) In Mininet, create a custom topology.
2) Configurate of all hosts, switches and links.
3) Initialize of the network.
4) Settings for load balancing algorithm in the controller.
5) Transmit hosts requests over the network.
6) Evaluate the performance of load balancing algorithm.

For the average bisection bandwidth, the comparison of experimental results is presented in Table 4. It's evident from the table that PSO algorithm is higher than ECMP and Round Robin in terms of average bisection bandwidth.

Table 4. Average Bisection Bandwidth (Mbps)

Algorithm	Value
ECMP	76
Round Robin	77
PSO	89

The delay results of using different algorithms to transmit different numbers of data packets are shown in Fig. 2. By comparing the results, it can be seen that compared with ECMP and round-robin technology, PSO has lower latency. This is caused by the fast convergence characteristic of the particle swarm algorithm towards the global solution.

Table 5 presents the average packet loss ratio for various techniques. It is evident that the round-robin technique exhibits a higher average packet loss ratio when compared to both ECMP and PSO.

In summary, the PSO algorithm has significant advantages in terms of average bandwidth, latency, and packet loss rate. Among them, the ECMP algorithm does not consider the current network state and only schedules traffic through hashing, it can easily lead to network congestion. Round robin also belongs to stateless scheduling. In this experiment, due to the same performance of each switch, its average bandwidth is better than ECMP. However, in actual networks, the performance load of each switch is different, which can easily result in load imbalance.

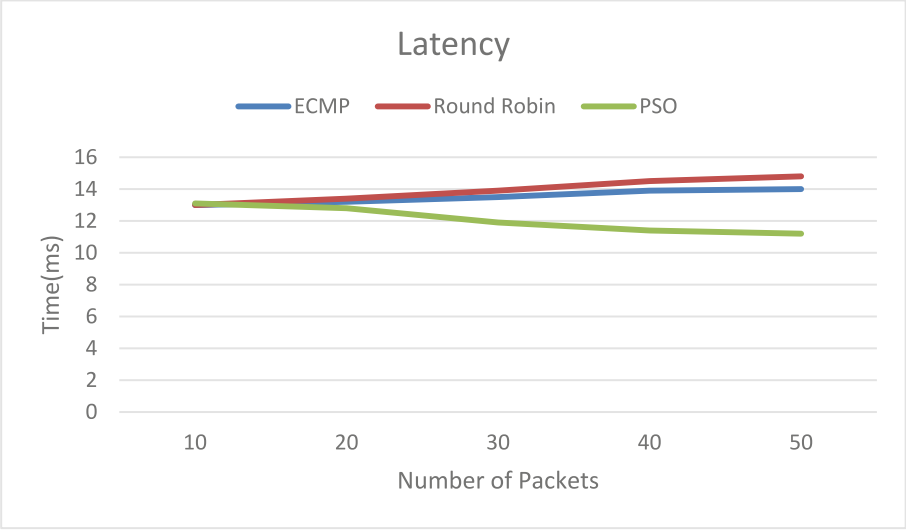

Fig. 2. Comparison of latency.

Table 5. Packet loss ratio

Algorithm	Value
ECMP	0.021
Round Robin	0.025
PSO	0.016

6 Summary

In this paper, we optimized load balancing in software defined networking (SDN) using particle swarm optimization (PSO). We evaluated the performance of PSO and compared with terms of bisection bandwidth, latency, and loss ratio of packet. The results indicate that particle swarm optimization algorithm can dynamically model based on load changes. However, reliability issues have not been addressed in this algorithm, and assuming the performance of all devices is the same, but in actual networks, the performance of devices is different.

In our future work, PSO algorithm must be appropriately improved, and evaluate load balancing while considering real-time networks, such as TCP and UDP.

Fund Project. Zhuhai College of Beijing Institute of Technology 2022 School-level Course (compute network) Beijing Institute of Technology Zhuhai College 2023 School -level Innovation and Entrepreneurship Training Project (Research and Implementation of Load Balancing and Energy-saving Technologies in SDN).

References

1. Kreutz, D., Ramos, F.M.V., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**, 14–76 (2015)
2. Abdelaziz, A., et al.: Distributed [3] controller clustering in software defined networks. PLoS ONE **12**, e0174715 (2017)
3. Akbar Neghabi, A., Jafari Navimipour, N., Hosseinzadeh, M., Rezaee, A.: Nature-inspired meta-heuristic algorithms for solving the load balancing problem in the software-defined network. Int. J. Commun. Syst. **32**, e3875 (2019)
4. Albowarab, M.H., Zakaria, N.A., Abidin, Z.Z.: Directionally-enhanced binary multi-objective particle swarm optimisation for load balancing in software defined networks. Sensors **21**(10), 3356 (2021)
5. Hamdan, M., et al.: A comprehensive survey of load balancing techniques in software-defined network. J. Netw. Comput. Appl. **174**, 102856 (2020)
6. Praveen, S.P., Sarala, P., Kumar, T.N.S.K.M., Manuri, S.G., Srinivas, V.S., Swapna, D.: An adaptive load balancing technique for multi SDN controllers. In: 2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS), Trichy, India, pp. 1403–1409 (2022)
7. Sridevi, K., Saifulla, M.A.: LBABC: distributed controller load balancing using artificial bee colony optimization in an SDN. Peer-to-Peer Netw. Appl. **16**, 947–957 (2023)
8. Chiesa, M., Kindler, G., Schapira, M.: Traffic engineering with equal-cost-multipath: an algorithmic perspective. IEEE/ACM Trans. Netw. **25**(2), 779–792 (2017)
9. Tanha, M., Sajjadi, D., Ruby, R., Pan, J.: Traffic engineering enhancement by progressive migration to SDN. IEEE Commun. Lett. **22**(3), 438–441 (2018)
10. Novaes, M.P., Carvalho, L.F., Lloret, J., Proenca, M.: Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. IEEE Access **8**, 83765–83781 (2020)
11. Mao, B., Tang, F., Fadlullah, Z.M., Kato, N.: An intelligent route computation approach based on real-time deep learning strategy for software defined communication systems. IEEE Trans. Emerg. Top. Comput. **9**(3), 1554–1565 (2021)
12. Xue, H., Kim, K.T., Youn, H.Y.: Dynamic load balancing of software-defined networking based on genetic-ant colony optimization. Sensors **19**, 311 (2019)
13. Zhu, S., Long, Y., Sun, G., Li, C.: Improved ant colony algorithm for network flow scheduling in SDN data center. J. Harbin Univ. Sci. Technol. **001**, 1–7 (2022)
14. Jamali, S., Badirzadeh, A., Siapoush, M.S.: On the use of the genetic programming for balanced load distribution in softwaredefined networks. Digit. Commun. Netw. **5**, 288–296 (2019)
15. He, X., Ren, Z., Shi, C., Fang, J.: A novel load balancing strategy of software-defined cloud/fog networking in the Internet of Vehicles. China Commun. **13**(Suppl. 2), 140–149 (2016)
16. Belgaum, M.R., Ali, F., Alansari, Z., et al.: Artificial intelligence based reliable load balancing framework in software-defined networks. Comput. Mater. Continuum **70**(1), 251–266 (2022). (in English)
17. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, no. 2 (1995)
18. Zhang, H., Lin, K.-Y., Huang, H.: Multi-objective scheduling model for OpenStack based cloud. In: 2021 8th International Conference on Computational Science/Intelligence and Applied Informatics (CSII) (2021)