An ACO-based Link Load-Balancing Algorithm in SDN

Chunzhi Wang, *Gang Zhang, Hui Xu and Hongwei Chen
Hubei University of Technology
School of Computer Science
Wuhan, China
square4@foxmail.com

Abstract—Software Defined Networking is a novel network architecture, which separates data and control plane by OpenFlow. The feature of centralized control to acquire and allocate of global network resource. So, the link load-balancing of SDN is not as difficult as the traditional network. This paper proposes a link load-balancing algorithm based on Ant Colony Optimization(LLBACO). The algorithm uses the search rule of ACO and takes link load, delay and pack-loss as impact factors that ants select next node. For maintaining link load-balancing and reducing end-to-end transmission delay, the widest and shortest path in the all paths can be gained by ants. The results of simulations show that LLBACO can balance the link load of network effectively, improve the Quality of Service(QoS) and decrease network overhead, compared with existing algorithm.

***Keywords*—Software Defined Networking, ACO, link load-balancing, QoS.**

I. INTRODUCTION

Software Defined Networking[1] possesses the features of centralized control, open interfaces and network virtualization, breaking a lot of limitations of traditional network and becoming the current focus of the industry, which has been widely used in the data center, WAN and other fields. B4[2],a private WAN connecting Google's data centers across the planet was transformed by SDN. It was a very successful large-scale SDN commercial case, which increased link bandwidth utilization rate from 30%~40% to nearly 100%. The success of B4 attracted more enterprises to join the tide of SDN. With the rise of cloud computing and big data, the increasing data need to be transmitted and disposed in the network. Different demand of business caused a huge difference in the load of each link and appeared the phenomenon that local load was unbalanced. So, the implementation of the link load-balancing strategy is indispensable to guarantee the stability of network operation.

Ant Colony Optimization(ACO) is a new heuristic algorithm with distributed computing, information positive feedback and heuristic search, has been widely used in various fields, such as load-balancing in NoC system[3],Bin Packing problem[4],Job Shop Scheduling[5],routing problem[6] and so on. In the process of solving the routing problem, all ants search new paths and collect states of the existing paths. If the ants succeeded in searching destination, then they would return to source along the original path. The ant colony could complete the task of path search after updating routing tables for all nodes based on the current path.

Multi-constrain routing is a NP-complete problem [7]. For simplifying the problem, LLBACO in the control plane makes full use of the mechanism that ant colony searches paths, takes link load as main factor, delay and pack-loss are the secondary factors. Thus, the algorithm can guarantee that the service flows forwarding on the widest and shortest path as much as possible.

The rest of paper is organized as follows. Section 2 shows some existing methods to solve load-balancing problem. We provides our link-balancing framework in Section 3. In Section 4, LLBACO is described in detail. Sections 5 analyzes the result of simulation. Section 6 concludes our study.

II. RELATED WORK

Plug-n-Server[8] is a network load-balancing system based on OpenFlow, which was proposed by the researchers of Stanford university. Controller in the system has three modules: network manager module, host manager module and flow manager module. Network manager collects the information about topology and link utilization. Host manager monitors the state and load of each server. Flow manager is an OpenFlow controller that manages and routes flows with load-balancing algorithm. The system has a strong guiding significance for the research of load-balancing in SDN.

LABERIO[9] is a dynamic load-balancing routing algorithm in OpenFlow network. When load detector finds the value of network load exceeds the threshold that was preset in the algorithm, the controller will distribute flows in the highly loaded links to other links. However, when controller receives a new request, the algorithm must compute all the paths between two nodes. This process is very time-consuming in a complex network. In 2015, Li-Der applied genetic algorithm [10] to the load-balancing problem in OpenFlow network. They firstly preset the flow tables in the switches. Once the flows outbreak or load increase suddenly on the servers, this algorithm would balance the load of links and servers. The use of genetic algorithm can save network cost and avoid the bottleneck of single controller. Dijkstra is a classic algorithm to solve shortest path problem. FSEM[11] is a path load-balancing solution based on SDN. It uses Top-K algorithm to select the shortest Top-K paths. Then, it chooses a best path

978-1-5090-3555-7/16 $31.00 © 2016 IEEE
DOI 10.1109/CCBD.2016.26

with fuzzy synthetic evaluation method in the Top-K paths. This scheme improves the performance of network overall, but Top-K algorithm only considers hop count, which may be some better path to be excluded.

III. Link Load-balancing Framework

The proposed link load-balancing system aims to balance the load of each link and avoid congestion. The framework of our solution for link load-balancing is showed in Figure 1. It consists of control plane and data plane. OpenFlow is the interface standard of the two planes. Then, the planes and the process of dealing with flows will be introduced in detail.

A. Control Plane

Control plane can control all devices in the data plane as the brain of software defined network. For realizing link load-balancing, our system mainly defines the following four modules in the controller.

1) Monitor module

SFlow is a network monitoring application used in traditional network. However, OpenFlow has many tuples, SFlow can only focus on one of the domains by sampling. When the network is abnormal, SFlow can not find faults in time. The monitor module makes network manager get the flow of visibility. Meanwhile, the manager can decide to monitor which flows.

2) Data collection module

Data collection module is able to count, store and analyze the information from data layer through the uplink channel of south interface supported OpenFlow. It is conducive to discover problem in the network and formulate corresponding strategy without delay.

3) Load-balancing module

Centralized control is one of the most important features of SDN, this module obviously embodies the feature .Whenever Packet-In message arrive to controller from switches, load-balancing module can compute the best path to destination according to the data from other modules. LLBACO is deployed in this module, will be described below.

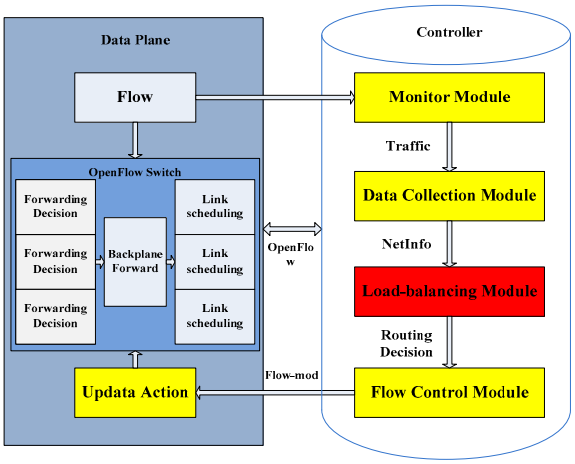

Fig. 1. Link Load-balancing Framework.

4) Load-balancing module

Flow table is a tool for controller informs switches how to deal with packets. After controller formulates forwarding strategy, flow control module sends new flow tables to switches by packaging the message named Flow-mod.

B. Data plane

The data plane of SDN concentrates on the function of fast-forwarding. So, it has a lower complexity compared the traditional network. The devices are uniformly controlled and managed through the OpenFlow protocol by the controller, so as to realize the virtualization of underlying devices.

C. Data flow processing

- When a data flow comes into data plane, switch at the entrance receives a packet and parses the packet header. The match starts with the first flow table. Flow entries match packets in priority order. If the match was successful, the instructions associated with the specific flow entry would be executed.
- If the match came to nothing, switch would take the packet into cache and send Packet-In to controller where decides how to deal with the packet.
- Data plane transmits the data flow to destination host along the path, which was computed by load-balancing module.

IV. Link Load-balancing ACO for SDN

A. Next Node Selection Strategy

In ant colony algorithm, roulette is the method that all ants select next site. As shown in Figure 2, the probability of ant on node 1 selects node 2 and node 3 are P(1,2) and P(1,3). The equation of P(i,j) is given as below.

$$P_{ij} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta} \cdot \mu_{ij}^{\gamma}}{\sum_{u \in allowSet} \tau_{iu}^{\alpha} \cdot \eta_{iu}^{\beta} \cdot \mu_{iu}^{\gamma}} & if\ j \in allowSet \\ 0 \end{cases} \quad (1)$$

The parameters α, β and γ represent the relative influence of each factor, allowSet is a set of nodes those ants can select, τ_{ij} is the pheromone intensity on link (i,j), $\eta_{ij}=1/Load_{ij}$, $\mu_{ij}=1/Cost_{ij}$, $Load_{ij}$ is the load of link (i,j), $Cost_{ij}$ is the transmission cost of link (i,j). They are all impact factors when ants select next node. $Cost_{ij}$ is then described as:

$$Cost_{ij} = \delta * delay_{ij} + (1-\delta)\, pl_{ij} \qquad 0 < \delta < 1 \quad (2)$$

where $delay_{ij}$ is the delay and pl_{ij} is the pack-loss on link (i,j).

B. Pheromone Updating Strategy

After ant arrives to destination, it will release pheromone along the same way. The pheromones always evaporate with time. This process is called pheromone updating. In this algorithm, the method of pheromone updating is global update, is that ant only leaves pheromone on the best path after all ants finish task. Pheromone updating is defined as:

$$\tau_{ij} = (1-\rho) * \tau_{ij} + \Delta\tau_{ij} \quad (3)$$

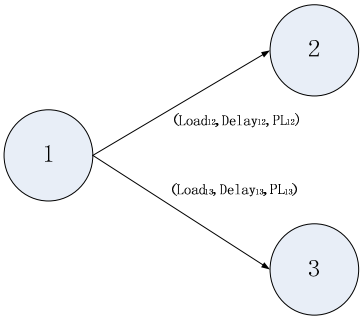

Fig. 2. Node Selection

Where,

$$\Delta\tau_{ij} = \begin{cases} \frac{Q}{L} & if\ (i,j) \in E \\ 0 \end{cases} \tag{4}$$

$\Delta\tau_{ij}$ is the amount of pheromone, which was left between node i and j by ants. $\rho \in (0,1)$, and it represents the speed of pheromone volatilization, Q is the total amount of pheromone, L is the length of the best path. Pheromone is the medium for ants communicate with each other, directly affecting the efficiency of ants search path.

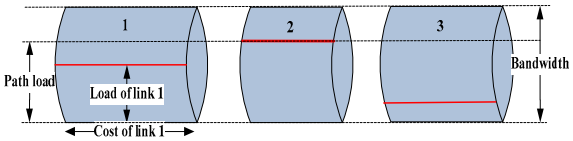

Fig. 3. Path Load and Cost

C. Calculating the Best Path

LLBACO is to balance link load and ensure the quality of network service. So, the best path based load, delay and pack-loss are selected. In a complex network, a low-load path is possible to have higher delay and pack-loss. For getting the best path, our algorithm sets a dynamic threshold and exclude those paths, the load value of which exceeds the threshold. Then, a minimum cost path is chosen in remaining paths.

1) Calculating path load and cost

Figure 3 shows a path which has three links with equal bandwidth, the height of red line represents the link load, the length of column represents the link cost. Obviously, the path load is the maximum link load, and the path cost is the sum of the cost of all links. The equation is as follows.

$$\text{Load}(path) = Max\{\text{Load}_{ij} \mid (i,j) \in path\} \tag{5}$$

$$\text{Cost}(path) = \sum_{(i,j)\in path} \text{Cost}_{ij} \tag{6}$$

2) Calculating threshold L_k and updating pathlist$_k$

pathlist$_k$ is the set of paths in the k th iteration. After working out the threshold L_k, new pathlist$_k$ can be gain by removing some paths of the load value is greater than L_k.

$$L_k = \frac{\sum_{path \in pathlist_k} \text{Load}(path)}{\text{Len}(pathlist_k)} \qquad k \le C \tag{7}$$

3) Getting the best path

As is shown in Equation 8,the best path is the minimum cost path in new pathlist$_k$.

$$BestPath = \arg_{path} \min\{\text{Cost}(path) \mid path \in pathlist_k, \text{Load}(path) \le L_k\} \tag{8}$$

D. Algorithm Description

LLBACO:Link load-balancing algorithm based ACO

```
LLBACO(G(V,E),src,dst)
begin
while (iter<M)
    for each ant k
        Set CurrNode= src
        while (CurrNode!=dst)
            Choose next node with P    /* Equation 1*/
            Path.append(CurrNode)
        end while
        PathList.append(Path)
    end for
    SelectBestPath(PathList)           /* Equation 8*/
    Update pheromone                   /* Equation 3*/
    BestPathList.append(tmpPath)
end while
SelectBestPath(BestPathList)
Print result;
end
```

The inputs of LLBACO are a given graph G(V,E). Each ant searches path based Equation 1.The algorithm uses PathList to store the path from the source node src to destination node dst. After the ant colony finishes a search, Equation 8 can compute the current best path, which is appended to the list named BestPathList. The pheromone will also be updated by Equation 3, then ant colony starts the next search. Finally, we can get the best path in the BestPathList.

V. Simulations and Results Analysis

A. Experimental Method

The advantage of our algorithm is that it can not only balance link load, but also improve QoS. So the experiment is to compare LLBACO with load-balancing based ACO (LBACO)[12] and round robin(RR) in the standard deviation of link load and transmission cost. When the standard deviation of link load and transmission cost are smaller, the effect of load-balancing and QoS are better. Standard deviation of link load is defined as follow.

$$LSD(linkload) = \sqrt{\frac{\sum_{i=1}^{n} (load_{link_i} - \overline{load_{links}})^2}{n}} \tag{9}$$

B. Experimental Result

The simulation used the network topology shown in Figure 4, which is composed of 6 nodes and 10 edges. We injected flows with a random number in 50M~100M and the total of injection was 200 times. We recorded the network load and cost after injecting flows. Table 1 is the data from simulation.

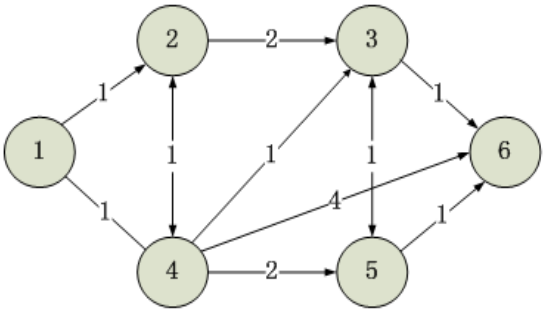

Fig. 4. Experimental Topology

TABLE I. DATA OF ALGORITHMS

Index \ Algorithm	LLBACO	LBACO	RR
Total Flow(M)	15895	16353	15951
Path Load (%)	0.7707	0.7447	0.7357
LSD	0.17	0.14	0.24
Total Cost	1057.653	1381.881	1806.518

From Figure 5, we can see that the Round-Robin has the biggest standard deviation of link load throughout the process, LBACO and LLBACO have a small difference. But, LBACO is an algorithm, which concentrates on load-balancing and ignores delay and pack-loss. So Figure 5 illustrates that the proposed algorithm has a good performance in balancing link load.

Then, we can get a comparison in the total transmission cost from Figure 6. The network used by LLBACO has a lower transmission cost compared with LBACO and RR. After 200 flow injections in our simulation, our algorithm can reduce transmission cost about 30% and 45% in comparison with LBACO and RR.

We extract the delay and pack-loss as reference data to express the performance of LLBACO in improving QoS. As is shown in the Figure 7 and Figure 8, LLBACO and LBACO have similar performances in the initial stage. However, with the increase of network load, the delay of LLBACO is stable at 8ms and pack-loss is 0.1%. But the delay and pack-loss of LBACO have violent fluctuations, which have a serious impact on QoS.

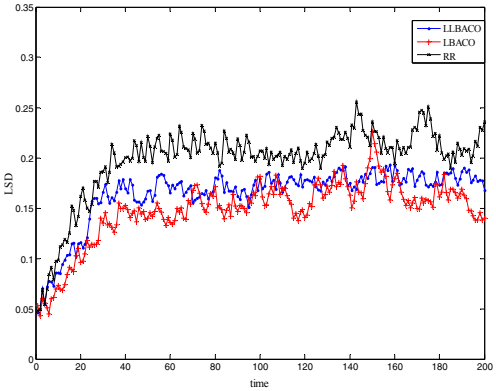

Fig. 5. Comparison in LSD of Algorithms

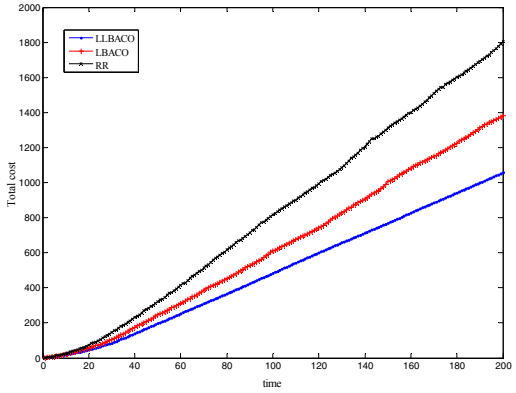

Fig. 6. Comparison in Total Cost of Algorithms

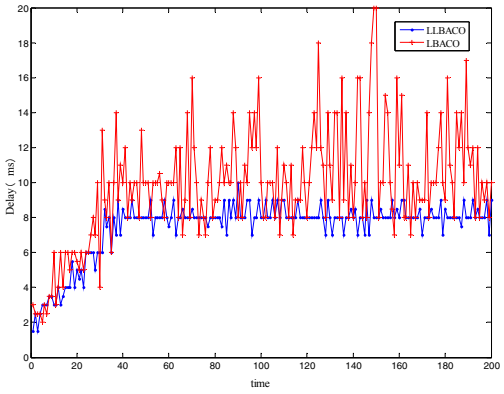

Fig. 7. Comparison in Delay of Algorithms

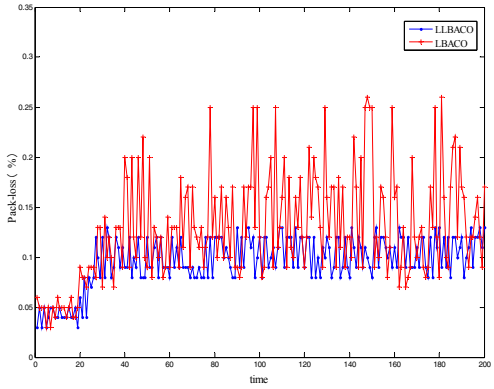

Fig. 8. Comparison in Pack-loss of Algorithms

VI. CONCLUSION

In this paper, we have presented a link load-balancing framework and the improved algorithm in detail. The proposed ACO based link load, delay and pack-loss narrows the search scope of path, simplifies multi-objective optimization problem by setting dynamic load threshold. The evaluation results shows that, the proposed method has a better performance in balancing network load and improving QoS, and then increases the stability and rapidity of network traffic forwarding.

ACKNOWLEDGMENT

This work has been supported by the General Program for National Natural Science Foundation of China (No. 61170135), National Natural Science Foundation of China (No.61602162), the National Natural Science Foundation of China for Young Scholars (No. 61202287), the General Program for Natural Science Foundation of Hubei Province in China (No. 2013CFB020).

REFERENCES

[1] Caokun Zhang, Yong Cui, Heyi Tang, Jianping Wu, "State-of-the-Art Survey of Software-Defined Networking," Journal of Software, vol. 26, no. 1, 2014, pp. 62-81.

[2] Sushant Jain, Alok Kumar, Subhasree Mandal, "B4: Experience with a Globally-Deployed Software Defined WAN," Acm Sigcomm Computer Communication Review, vol. 43, no. 4, 2013, pp. 3-14.

[3] Hsien-Kai Hsin, En-Jui Chang, Chi-hao Chao and An-Yeu Wu, "Regional ACO-based routing for load-balancing in NoC systems," Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress on. IEEE, 2010, pp. 370-376.

[4] Oscar D. Lara, Miguel A. Labrador, "A multiobjective ant colony-based optimization algorithm for the bin packing problem with load balancing," Evolutionary Computation (CEC), 2010 IEEE Congress on. IEEE, 2010, pp. 1-8.

[5] Rajesh Chaukwale, Sowmya Kamath S, "A Modified Ant Colony Optimization Algorithm with Load Balancing for Job Shop Scheduling," Advanced Computing Technologies (ICACT), 2013 15th International Conference on IEEE, 2013, pp. 1-5.

[6] Peter Janacik, Dalimir Orfanus, Adrian Wilke, "A Survey of Ant Colony Optimization-Based Approaches to Routing in Computer Networks," 14th International Conference on Intelligent Systems, Modelling and Simulation, 2013, pp. 427-432.

[7] Xuxun Liu, Desi He, "Ant colony optimization with greedy migration mechanism for node deployment in wireless sensor networks," Journal of Network & Computer Applications, vol. 39, no. 1, 2014, pp. 310-318.

[8] Handigol N, Seetharaman S, Flajslik M, et al. "Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow," In: Proceedings of demo at ACM SIGCOMM, vo1.4, no. 5, 2009.

[9] Hui Long, Yao Shen, Minyi Guo, Feilong Tang, "LABERIO: Dynamic load-balanced Routing in OpenFlow-enabled Networks," IEEE International Conference on Advanced Information Networking & Applications. IEEE, 2013, pp. 290-297.

[10] Li-Der Chou, Yao-Tsung Yang, Yuan-Mao Hong, Jhih-Kai Hu, Bill Jean, "A Genetic-Based Load Balancing Algorithm in OpenFlow Network," Advanced Technologies, Embedded and Multimedia for Human-centric Computing. Springer Netherlands, vol. 260, 2014, pp. 411-417.

[11] Jun Li, Xiangqing Chang, Yongmao Ren, Zexin Zhang, Guodong Wang, "An Effective Path Load Balancing Mechanism Based on SDN," Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on. IEEE, 2014, pp. 527-533.

[12] Huanqing Zhang, Xueping Zhang, Haitao Wang, Yanhan Liu, "Task Scheduling Algorithm Based on Load Balancing Ant Colony Optimization in Cloud Computing," MICROELECTRONICS &COMPUTER. vol. 32, no. 5, 2015, pp. 33-44.