sensors

MDPI

Article

Dynamic Load Balancing of Software-Defined Networking Based on Genetic-Ant Colony Optimization

Hai Xue [1], Kyung Tae Kim [2] and Hee Yong Youn [1,*]

[1] Department of Electronic, Electrical and Computer Engineering, Sungkyunkwan University, (16419) 2066, Seobu-Ro, Jangan-Gu, Suwon-Si, Gyeonggi-Do, Korea; xuehai@skku.edu

[2] College of Software, Sungkyunkwan University, (16419) 2066, Seobu-Ro, Jangan-Gu, Suwon-Si, Gyeonggi-Do, Korea; kyungtaekim76@gmail.com

* Correspondence: youn7147@skku.edu; Tel.: +82 31-290-7147

Received: 15 December 2018; Accepted: 9 January 2019; Published: 14 January 2019

Abstract: Load Balancing (LB) is one of the most important tasks required to maximize network performance, scalability and robustness. Nowadays, with the emergence of Software-Defined Networking (SDN), LB for SDN has become a very important issue. SDN decouples the control plane from the data forwarding plane to implement centralized control of the whole network. LB assigns the network traffic to the resources in such a way that no one resource is overloaded and therefore the overall performance is maximized. The Ant Colony Optimization (ACO) algorithm has been recognized to be effective for LB of SDN among several existing optimization algorithms. The convergence latency and searching optimal solution are the key criteria of ACO. In this paper, a novel dynamic LB scheme that integrates genetic algorithm (GA) with ACO for further enhancing the performance of SDN is proposed. It capitalizes the merit of fast global search of GA and efficient search of an optimal solution of ACO. Computer simulation results show that the proposed scheme substantially improves the Round Robin and ACO algorithm in terms of the rate of searching optimal path, round trip time, and packet loss rate.

Keywords: load balancing; Software Defined Networking; genetic algorithm; Ant Colony Optimization; genetic-Ant Colony Optimization

1. Introduction

The traditional Internet Protocol (IP) networks need to be enhanced to efficiently deal with huge volume of network traffic generated due to the rapid growth of Internet of Things (IoT). As an outcome of the search for an efficient solution for this issue, a new type of networking paradigm called Software-Defined Networking (SDN) has been proposed [1]. SDN changes the way of traditional network management by breaking the vertical integration of network components, separating the network control logic from the underlying routers and switches, promoting (logical) centralization of network control, and introducing the ability of programming the network operation. OpenFlow is the most widely used communication protocol between the controller plane and switch plane in SDN.

As aforementioned, SDN is quite different from the traditional IP network, and several issues need to be resolved including data forwarding, load balancing (LB), energy management, and so on. In SDN, flow tables are widely used of which entry consists of some fields including the header, counter, and action. When a packet arrives at a switch, lookup of the flow entries is carried out. The header and counter of a network flow are updated when it is changed for some reason such as LB or rerouting. The switch processes the data flow based on the header information and preset rules to control the network traffic. The flow control is based on the algorithm employed to balance the traffic

loads [1]. Due to its inherent versatile nature of SDN, dynamic LB is very important for the centralized controller of SDN.

There exist various studies on LB of SDN. Kang et al. [2] presented a scheme on how to use the genetic algorithm (GA) for solving the LB problem of SDN. Even though it is an innovative approach, the classical GA is highly time-consuming. There are also studies adopting Ant Colony Optimization (ACO) for LB [3–5]. In Li et al. [3], a traffic-engineering framework was proposed which contains a heuristic algorithm layer and meta-layer based on machine learning. For the traffic engineering, the heuristic algorithm layer is trained by the meta-layer to find an optimal path after training. Then, dynamic routing occurs via the optimal path, resulting in efficient network operation. ACO was also employed in QoE-Centric flow routing [4], which was shown to be better than Shortest Path Routing (SPR). In Lin et al. [5], ACO was combined with job classification for multi-controllers, where job classification distinguishes central controllers from sub-controllers. Even though the basic ACO algorithm renders good results, a lot of computations are needed before it is fully operational. As with the typical optimization problems, the solution may take a long time to converge or lead to the local optimum. It is also not easy to be deployed to the network when the operation scenario changes.

In this paper a novel approach for LB of SDN is proposed which combines GA with ACO, called Genetic-Ant Colony Optimization (G-ACO). The existing scheme based on the ACO algorithm employs the positive feedback mechanism for updating the path information of the flows when they are forwarded. However, this may result in a local optimal solution and improper selection of the path. The random selection policy can be employed to avoid the local optimal solution, but it does not guarantee even suboptimal solution. These issues limit the performance of ACO in the path selection of SDN. The trade-off between local optimal solution and random selection is still an issue with ACO. The GA is thus adopted with ACO to alleviate the problem and enhance the LB of SDN. The GA is applied in the second stage of the search to effectively reduce the search space, whereupon the ACO algorithm can efficiently find the paths of the flows for LB. Computer simulation reveals that the proposed scheme substantially improves the roundtrip time (RTT) and packet delivery ratio compared to the Round Robin (RR) and ACO algorithm by effectively achieving the LB.

The rest of the paper is structured as follows. Section 2 provides an overview of the ACO and GA, and LB for SDN. In Section 3 a new method for dynamic LB of SDN is proposed. Also, an analytic model of the proposed scheme is presented. Section 4 gives the performance evaluation of the proposed scheme with the controller, OpenDayLight, and switch emulator, Mininet. Finally, Section 5 concludes the paper and outlines future research direction.

2. Related Work

In this section SDN, GA, ACO, and LB are briefly discussed. LB is a crucial problem with SDN, while GA and ACO are the algorithms which are adopted in the proposed scheme for solving the problem.

2.1. SDN

SDN is a new technology in the field of computer networking, which is presently receiving a great deal of attention. It originated from a project in academia [5]. In SDN, the network control making decisions on traffic routing (control plane) is decoupled from the part forwarding the traffic to the destination (data plane). The network is directly programmable, and the infrastructure is allowed to be abstracted for the applications and network services. The experts and vendors claim that this greatly simplifies the networking task [6]. Here, balancing the traffic load of the switches is a very important issue.

LB has been extensively studied for traditional IP network. Recently, some studies were conducted on SDN. Since SDN decouples the control plane and data plane of the network, LB between the controller and switches directly affects the stability of the whole network. The controller running on

the control plane controls data forwarding by managing the flow tables of switches. The flow table consists of the flow entry, and each flow entry mainly consists of six fields as listed in Table 1 [7].

Table 1. The fields of a flow entry in the flow table.

Field	Description
Match	Port, packet header, and metadata forwarded from the previous flow table
Priority	Matching precedence of the entry
Counter	Statistics for matching the packets
Instruction	Action or pipeline processing
Timeout	Maximum effective time or free time before the entry is overdue
Cookie	Opaque data sent by the OpenFlow controller

Among several protocols proposed for SDN, the OpenFlow protocol [8] is the most commonly used one for the implementation of SDN. The main feature of OpenFlow is the programmability. By programming the controller, one can easily achieve the required network functionality. There is a total of three layers in the entire infrastructure of SDN. On top of the control plane is the application layer and below is the data forwarding layer. The structure of SDN based on the OpenFlow protocol is shown in Figure 1.

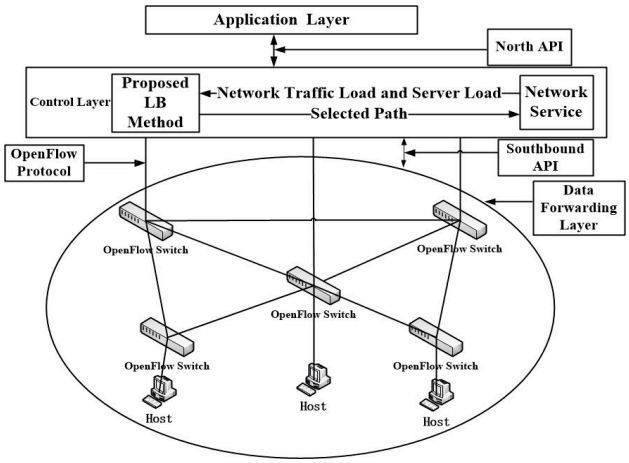

Figure 1. The structure of SDN based on the OpenFlow protocol.

2.2. ACO

The ACO algorithm employs a probabilistic approach for solving the target problem such as GA, simulated annealing, and so on. All such algorithms are based on some heuristics. The ACO algorithm was inspired by the behavior of ants. Initially, all the ants wander aimlessly, but they return the nest after laying down pheromone on the trails while searching for food. Other ants then can follow the pheromone trails instead of travelling randomly. This approach has been adopted to solve the problem of computer networking such as finding the best path in the network. There are various schemes based on the ACO algorithm such as AntNet, AntHocNet, HopNet and Stigmetry, which can be applied to solve the routing problem for computer network [9,10].

The ACO algorithm works in two steps. In the first step, the preceding ants discover new possible routes as well as gather information on the existing paths, which are referred to as forward update. If one of them successfully reaches the destination node, in the second step, some ants are sent back to the source node along the path previously explored by the preceding ant. During the backward trip, the routing tables of the nodes along the path are updated called backward update. With SDN, the forward update and backward update can be eliminated since the controller has the global view of the entire network [11].

2.3. GA

As aforementioned, GA is also a heuristic algorithm based on the evolutionary steps of natural selection and genetics. It is commonly used to generate high-quality solutions for the optimization and search problems using bio-inspired operators such as mutation, crossover, and selection [12]. Some researchers explored an intelligent exploitation of random search used to solve the optimization problem involving global search of a large space. Generally, there exist five steps in GA for solving a problem; initial population, fitness function, selection operation, crossover operation, and mutation operation. In the proposed scheme, the five steps are also adopted as a supplement to ACO in the second stage of the search of the path. It applies selection, intercross and mutation on the solution of each generation to broaden the search domain and reduce the search time of optimal path.

2.4. LB of SDN

There exist two ways of classifying the LB approach of traditional network. One is based on the object to which LB is applied; hardware LB and software LB. With hardware LB, LB is applied to individual hardware device, and thus its effect on the network operation is more direct and faster than software LB. However, it is costlier due to the involvement of hardware implementation. Another way of classification is static LB and dynamic LB, determined based on the adaptability of the traffic management policy. With static LB, the performance parameters are analyzed at the start of the network operation, and then the LB policy is determined based on that. It is easy to deploy and not costly, but the performance is low. With dynamic LB, the LB policy is dynamically updated based on the current condition of the network to achieve higher performance. However, it usually involves several steps of operations and the cost is higher. Due to the versatile characteristics of SDN, the software LB and dynamic LB policy are adopted in the proposed scheme.

Various LB algorithms have been developed for evenly distributing the traffic to the nodes in the network including the RR algorithm, Weighted Round Robin (WRR), Least-Connection (LC), Weighted Least-Connections (WLC), Fastest Response-Time (FRT) algorithm, and so on. With the RR algorithm, a round link table is managed, which holds the information on the nodes of the network. The tasks are forwarded to each node by circular turns so that each node gets the task equally likely. This algorithm is easy to implement, but it does not consider the condition of the nodes and thus congestion may easily occur. Unlike the RR algorithm, the WRR algorithm assigns a weight value to each node which is determined by the specification parameter of the nodes. A large weight means higher processing capability. WRR is obviously more effective than RR.

The LC algorithm assigns a new connection request to the node of least connection since the load condition of a node is estimated based on the number of connections. The device managing LB needs to keep and update the number whenever a new connection is made or one is disconnected. The WLC algorithm uses the weight with the LC algorithm. The node of higher weight is assigned more connections. The FRT algorithm analyzes the load condition of a device by sending a probe request to it (for example ping). Here, the server responding to the probe fast is deemed to be less busy. Since it is not the response time to an actual task but a probe, it may not be able to deliver the real load condition. We next present the proposed scheme combining GA with ACO, the G-ACO scheme, which capitalizes the key properties of SDN for LB.

3. The Proposed Scheme

In this section, first, the way the basic ACO algorithm is used for the LB of SDN is introduced along with its shortcomings. Then, the proposed LB scheme based on G-ACO is presented.

3.1. LB with ACO

In order to illustrate the concept of ACO employed in the proposed scheme, the topology shown in Figure 2 is used. Each ant in the ant-world can be viewed as a packet transmitted in the network, and it is started randomly at any node of the topology.

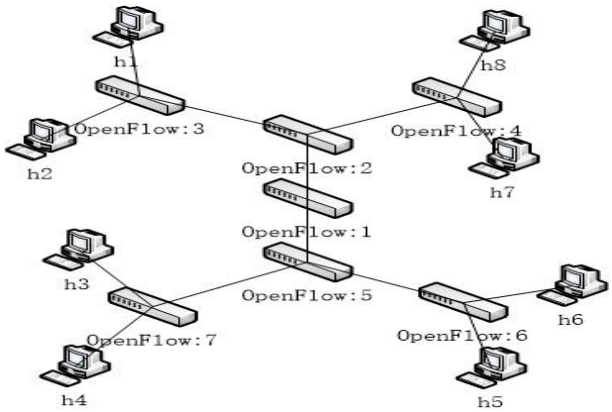

Figure 2. The test topology.

Assume that m and n are the total number of ants and switches, respectively, and $b_i(t)$ represents the number of ants in Switch-i, S_i, at time t. Then,

$$m = \sum_{i=1}^{n} b_i(t) \tag{1}$$

Each ant, A_k (k = 1,2, . . . , m), bears a taboo table to avoid visiting the same switch again, which holds the switches already passed by. It also calculates the state transition probability between two switches based on the remained pheromone on each path. They select the next switch to visit depending on the calculated probability. Denoting $P_{ij}^{k}(t)$ as the probability of A_k to visit S_j from S_i [13], then:

$$P_{ij}^{k} = \begin{cases} \frac{\left[\tau_{ij}(t)\right]^{\alpha} \cdot {\eta_{ij}}^{\beta}}{\sum_{a_k}\left[\tau_{ij}(t)\right]^{\alpha} \cdot {\eta_{ij}}^{\beta}}, & j \in a_k \\ 0 \qquad\qquad Others \end{cases} \tag{2}$$

Here a_k represents the switches available to be selected when A_k is in S_i. $\tau_{ij}(t)$ denotes the total pheromone laid on the path from S_i to S_j. $\tau_{ij}(0) = \delta$ indicates that the amount of pheromones on path is initially δ. α and β are the weight of the remaining pheromone and path distance in making the selection, respectively. η_{ij} is a heuristic function defined as:

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{3}$$

Here, d_{ij} represents the distance between S_i and S_j. Therefore, the probability of visiting S_j from S_i becomes higher as d_{ij} gets smaller. The two issues with the current mechanism of ACO in selecting the next switch are as follows.

- If α is small, it depends mainly on the value of d_{ij}.
- If β is small, it is selected randomly.

The first issue may cause the local optimal solution, while the second one does not guarantee finding any good solution. To resolve the issue of too much remaining pheromone, an update algorithm was proposed considering the behavior of real ants in nature [14]. Here, the pheromone of a path is

updated when an ant passes a switch (partial renewal) or whole path (global renewal), and the update rule is as follows.

$$\tau_{ij}(t+n) = (1-\rho) * \tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (4)$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \quad (5)$$

Here ρ is the percentage of pheromone volatilization on the path, and thus $(1-\rho)$ represents the portion of remaining pheromone. $\Delta{\tau_{ij}}^{k}(t)$ is the amount of pheromone increased by A_k on the path from S_i to S_j in one cycle. $\Delta{\tau_{ij}}^{k}(0)$ is 0 as it represents the initial state of each path.

In the process of searching an optimal path with ACO, each ant selects the next switch based on the probability of Equation (2). Meanwhile, a positive feedback mechanism is adopted to assign more weight to the current optimal path. In other words, the information on the traversed path is given more weight after each iteration. This approach can easily cause local optimum. If the random selection policy is employed to avoid local optimum, even a good solution is not guaranteed to be found or it may take long time. These issues are resolved by the proposed scheme explained next.

3.2. The Proposed G-ACO Scheme

Delay time and packet loss rate are the two criteria evaluating the Quality of Service (QoS) of the network [15]. To deal with the limitation of the ACO algorithm applied to the SDN for LB, the G-ACO algorithm is proposed. As in the typical GA, it involves selection, crossover, and mutation operation. The motivation is to enhance the rate of convergence to optimal LB and the capability of finding the global optimum. The delay time in sending packets and the packet loss rate will then be decreased. It is known that GA is effective for the global search of a large space. However, feedback information cannot be used and a lot of redundant iterations occur with a certain search space. The efficiency of finding the solution is also very low. On the other hand, ACO employs a positive feedback mechanism. However, in the second stage, the searching speed is very low due to the limited pheromone on the paths. The proposed G-ACO algorithm capitalizes the merit of the GA and ACO such that GA is used to properly distribute the pheromone and then the ACO algorithm is used to seek the solution. It lets the two algorithms complement each other for effective LB in SDN. The larger the network and the longer it has been operated, the more the proposed G-ACO scheme will decrease the RTT and packet loss rate using the accumulated information compared to the existing schemes. The steps of G-ACO algorithm are as follows.

- Step 1. Initial population

With G-ACO, the path is encoded for searching the path which involves integral number. For example, assume that a packet reaches a host in one iteration through several switches, which are S_7, S_5, S_1, S_4, and S_8. Then, the code of the path of this packet is (7,5,1,4,8).

One path is generated with each ant in one iteration. Since there are m ants, m paths are obtained, which are selected as the initial population.

- Step 2. Fitness function

There are several factors that need to be considered for deciding the fitness function of GA, including the length of path, energy consumed for receiving or sending a packet in each switch, energy status of the whole network, and so on. In this paper the length of the path is considered as the primary factor. The fitness function for path-p is then as follows:

$$f(p) = W \times l(p) \quad (6)$$

Here W is the weight value of the length and $l(p)$ is the length of the path. Note that the path of smallest $f(p)$ is selected.

- Step 3. Selection

In the selection process, the code of a path is obtained after each iteration of search. The load is calculated by the fitness function, and then the optimal one is selected for the next iteration. After several selection operations, the pheromone on the optimal path will be larger than the others, indicating that it will be more likely to be selected. As a result, the speed for searching the optimal path is increased.

- Step 4. Crossover

In order to avoid the situation of stagnation during the search operation, the crossover operation is adopted. It can expand the search scope and avoid the local optimum. After an iteration of search by ACO, some sub-optimal paths and optimal path might be obtained. Then, the crossover operation is applied to them. The aim of this step is to include more candidate paths leading to the optimal one.

Here, the crossover operation proposed by [16] is used such that the crossover operation is performed with the predefined crossover probability. The subsequence of the path of the offspring is determined from the subsequence of the paths of the parent. For example, assume that two parents, P_1 and P_2, are given:

$$P_1 = (9,8|\,7,6,5|\,4,3,2) \quad P_2 = (2,4|\,7,5,8|\,6,9,3) \tag{7}$$

In order to get the sequence of the offspring by crossover operation, first the cross segment is copied to the sub-generation, g_1 and g_2, as:

$$g_1 = (**|\,7,6,5|\,***) \qquad g_2 = (**|\,7,5,8|\,***) \tag{8}$$

Then, the coincident character sequence needs to be deleted in both P_2 and g_1. For example, '7', '6', '5' of g_1 is deleted from P_2, which leaves (2,4| 8| 9,3) in P_2. After the deletion, the remaining sequence of P_2 is copied to get g_1 = (2,4| 7,6,5| 8,9,3). Similarly, g_2 = (9,6| 7,5,8| 4,3,2) is obtained.

In this paper P_1 is assumed to be the optimal path, while P_2 is sub-optimal. Procedure 1 of crossover operation is as follows.

Procedure 1. Crossover operation of G-ACO

1: Assume that the path of P_1 is (x_1, y_1, z_1) and that of P_2 is (x_2, y_2, z_2). The crossover operation occurs with y_1 and y_2, and one of the new paths obtained is P_3: (x_1, y_2, y_1, z_1).
2: Deleting the duplicated switch, the new path P_3 is determined.
3: By the same way, another new path P_4 is obtained.
4: Applying the fitness function to P_1, P_2, P_3, P_4, an optimal path is selected

- Step 5. Mutation

Mutation operation is based on the predefined mutation probability. Two points from the paths of the offspring are randomly selected for mutation operation. For example, for the given g_1,

$$g_1 = (2,4|\,7,6,5|\,8,9,3) \tag{9}$$

After the mutation operation of exchanging '5' and '3', g_1 becomes:

$$g' = (2,4|\,7,6,3|\,8,9,5) \tag{10}$$

Procedure 2 of mutation operation is given below.

Procedure 2. Mutation operation of G-ACO

1: The mutation occurrence is based on the frequency which is defined in the simulation part, and the number of switches in the optimal path is m.
2: Randomly generate two natural numbers, n_1 and n_2 ($n_2 < n_1 < m$).
3: By exchanging the switch at the location of n_1 and n_2 of the optimal path, P_0, a new path, P_n is obtained.
4: Obtain the fitness of P_0 and P_n, and the one of the smaller value is selected as the optimal path

After the operations of G-ACO, the optimal path is determined and packets are transmitted along it. The basic steps of the proposed G-ACO are summarized in Procedure 3, and the flowchart of the proposed G-ACO Algorithm is shown in Figure 3.

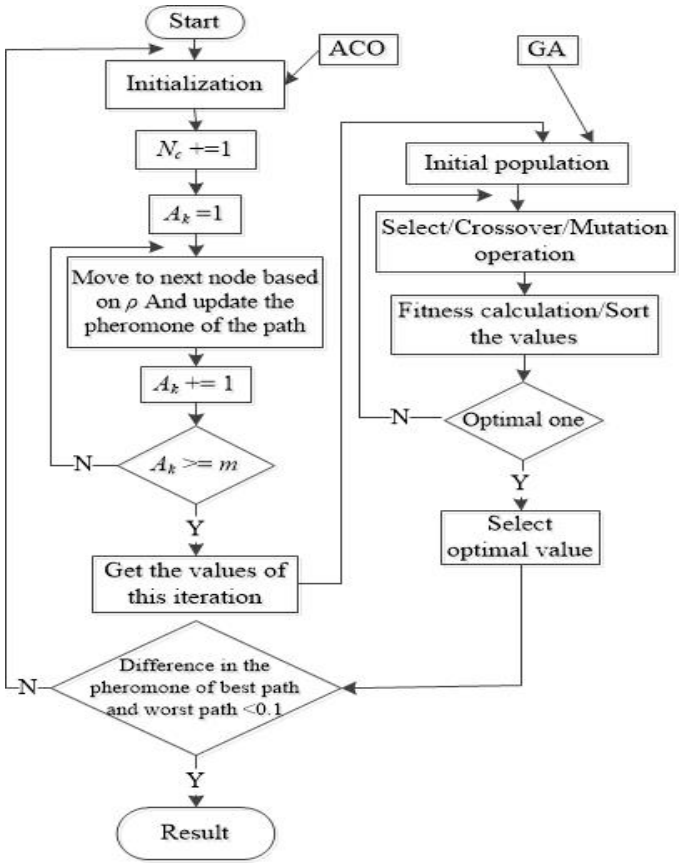

Figure 3. The flowchart of the proposed G-ACO.

Procedure 3. LB with G-ACO

1: N_c + = 1; (Iteration times)
2: A_k = 1;
3: A_k selects the next switch based on the calculated probability of Equation (2); meanwhile, update the taboo table and path pheromone.
4: k + = 1;
5: If $k \leq m$ (the total number of ants), go to Step 3, otherwise execute the GA;
6: Obtain the fitness value, and if the value satisfies the constraint, exits the loop. Otherwise, jump back to Step 1

4. Performance Evaluation

In this section an experiment is conducted to evaluate the performance of the proposed scheme. Also, it is compared with the RR and ACO scheme used for LB of SDN to verify its effectiveness.

4.1. Environment of Experiment

There exist separate tools used for the SDN controller and switch as the control plane and data plane are decoupled. In this paper OpenDayLight and Mininet are employed for setting up the controller and switch, respectively. First, a brief introduction of these tools is given.

Mininet is a network emulator creating a network of virtual hosts, switches, controllers, and links. It hosts standard Linux network software, and the switches employ OpenFlow for flexible routing with the SDN. Mininet supports research, development, learning, prototyping, testing, debugging, and many other tasks that could benefit from having a complete experimental network on a laptop or PC [17]. Some key features of Mininet are listed below.

- Supports relevant protocols of SDN such as OpenFlow
- Supports open Python API for developers
- Supports relevant modules of SDN such as Open vSwitch
- Highly scalable
- Developers can customize the topologies as needed
- Supports co-development among several engineers

OpenDayLight [18] is a collaborative open source project hosted by the Linux Foundation. The goal of the project is to accelerate the adoption of SDN and create a solid foundation for Network Functions Virtualization (NFV). It is written in Java, and the main modules of OpenDayLight are as follows:

- Topology Manager: Responsible for the entire network topology
- Forwarding Rule Manager: Manages the actions of entire network by adding, searching, deleting, and updating the flow rules
- Service Abstract Layer: Core module of OpenDayLight, abstracting some parts of the network by southbound interface and supporting application layer
- Host Tracker: Manages the information of the hosts by keeping the IP and MAC address, and establishes and deletes the connections to northbound interface
- Stats Manager: Manages the whole information of the network

In this experiment OpenDayLight and Mininet are installed separately on two PCs. The parameters of the PCs are listed in Table 2.

Table 2. The specification of the PCs.

	PC_1	PC_2
OS	Windows 10	Ubuntu 16.04
Virtual Machine	VMware Workstation 12 (OS: Ubuntu 16.04)	None
CPU	i3-4150	i3-4350
RAM	8G	8G
Hard Disk	500G	500G

4.2. Test Topology

At first, some flow tables were sent to the specified switch with the topology of Figure 2 to verify proper connection of OpenDayLight and Mininet. Some flow tables were sent to S_5 (OpenFlow:5), and the flow table delivery was verified as the snapshot shown in Figure 4.

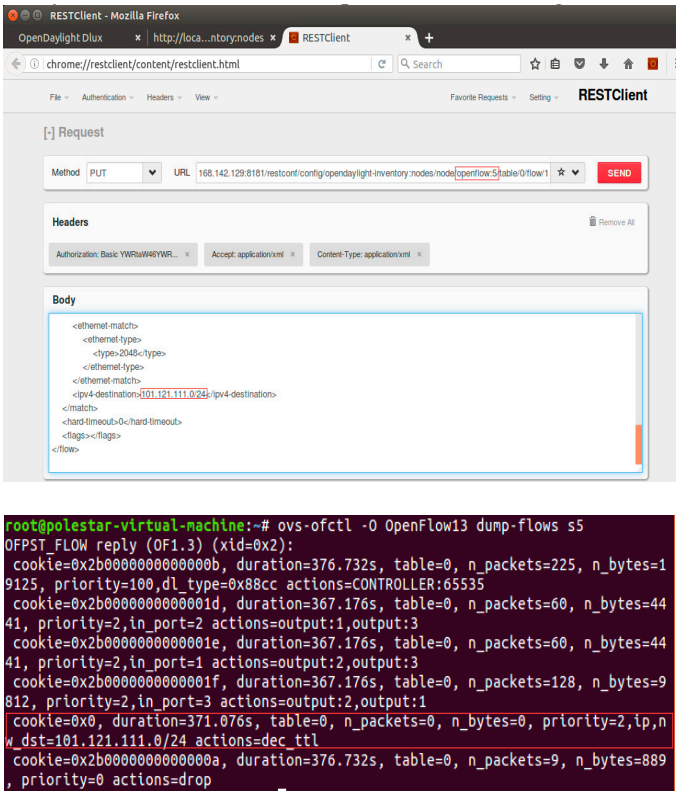

Figure 4. The result of flow table delivery.

A fat-tree topology is adopted to verify the effectiveness of the proposed LB scheme. The fat-tree topology has numerous advantages as elaborated in [19,20], and the target fat-tree topology consisting of 1 controller and 10 switches is shown in Figure 5.

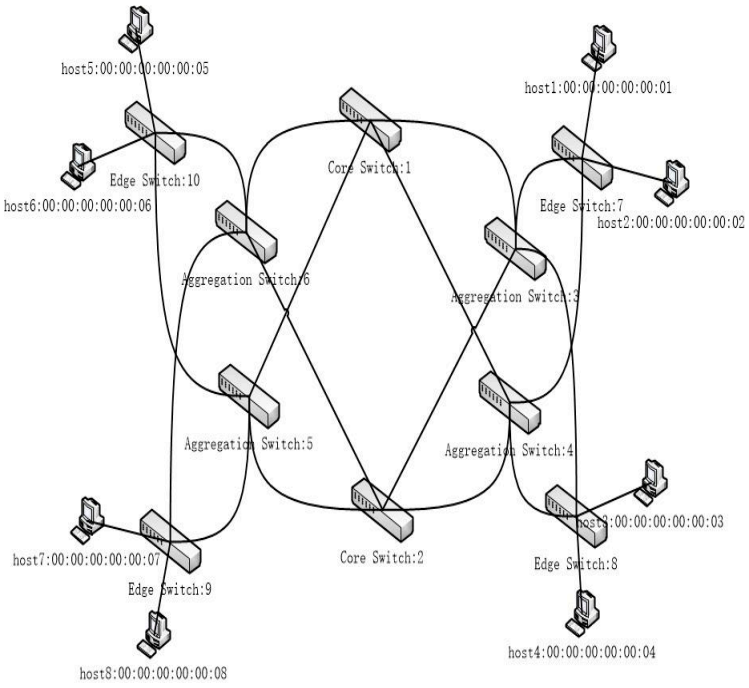

Figure 5. The target fat-tree topology.

4.3. Simulation Results

For fair comparison of the proposed scheme with the existing schemes, the Ant-Cycle method is employed for pheromone update [21]. The update rule and increment of pheromone are based on Equations (4) and (5), respectively. However, $\Delta\tau_{ij}{}^{k}(t)$ is modified as follows:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L_k} & \text{if } A_k \text{ moves from } S_i \text{ to } S_j \text{ in one iteration} \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

Here, L_k is the total length of A_k for finishing a round trip. As both the ACO and GA have the characteristics of astringency, the proposed G-ACO scheme may also have similar property. The factor ρ in Equation (4) can affect the search capability and the speed of astringency of the whole network. If ρ is too large, the volatilization rate of pheromone is high, leading to random selection. On the contrary, if it is too small, the volatilization rate will be low, resulting in local optimum. Therefore, selecting a proper value of ρ is very important. In the experiment, m (the number of ant) = 6, α (pheromone weight) = 1, β (distance weight) = 2 and Q (constant) = 1, which are determined empirically. The numbers of iteration steps allowing optimal path are listed in Table 3 as ρ is varied.

Table 3. The number of iterations with different ρ.

Pheromone Volatilization Factor (ρ)	Iterations
0.1	3
0.3	7
0.5	8
0.7	13
0.9	28

Notice from Table 3 that the number of iterations increases when ρ is raised. This implies that the amount of pheromone on a path significantly delays the path selection. On the contrary, when ρ is as low as 0.1, the amount of pheromone on the path imposes too much influence on the path selection. Referring to Table 3, the reasonable range for ρ seems to be 0.3~0.5, and thus the value of ρ is set to be 0.4 in the experiment.

In Equation (2) α and β are important parameters of G-ACO for an ant to select a path. α determines the dependency on the pheromone. If α increases, the probability for an ant to follow the path taken by a previous one will be higher. Then, the random nature of path selection will become smaller, leading to local optimum. On the contrary, β is the factor for randomness in path selection. If β increases, the probability for an ant to randomly select the next path becomes higher. Then, an optimal path may not be found in reasonable amount of time. Therefore, a simulation was carried out to determine a proper value of α and β. First, the value of α and β is set empirically. Then, α is changed while β is fixed. Similarly, β is changed while α is fixed. The simulation results are shown in Table 4, which show that the number of iterations decreases while α and β get larger. Here, α and β are set to 1 and 2, respectively.

Table 4. The number iterations taken to find an optimal path.

α	B	Iterations
0.1	0.1	33
0.1	0.5	17
0.5	0.5	8
1	2	7
3	7	3
5	9	2

In the proposed G-ACO scheme there is another important factor which is the number of ants, m. If it is too small, the search of the whole network will be limited, leading to local optimum. If it is too large, however, the effect of positive feedback diminishes in path selection, resulting in random path selection. Considering the importance of the number of ants, simulation has been carried out again to determine the number. The simulation result is shown in Table 5. Considering the convergence and randomness of G-ACO, m is set to 8.

Table 5. The number of ants and iterations.

M	Iterations
2	20
4	12
6	10
8	9
10	2

In summary, the parameters are set as $\rho = 0.4$, $\alpha = 1$, $\beta = 2$, $m = 8$. For the GA, the probability of crossover and mutation are set to be 0.9, and 0.04, respectively. Finally, the convergence condition, which is the difference in the amount of pheromone on the largest and smallest path, is set to 0.1.

Figure 6 shows the success rate of finding the optimal path with the proposed G-ACO, ACO and GA scheme. An optimal path from the initial node to the end node can be determined by the calculation of the fitness function. Then, a path is obtained with the respective scheme after each simulation run. By the comparison with the optimal path, it is judged whether it is an optimal path or not. A 10-minute simulation time is executed for each of the three schemes, and the success rates are compared in Figure 6. Observe from the figure that the proposed G-ACO scheme allows an approximately 95% success rate which is much higher than the other two schemes.

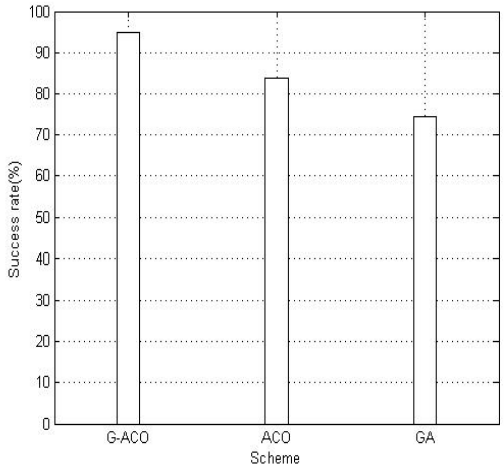

Figure 6. The comparison of success rates.

In the target network, the bandwidth of each load is assumed to be 100 Mbps. The iperf software tool [22] is used for the simulation run for 5 min and 10 min. The RR, ACO and the proposed G-ACO are simulated, RTT and packet loss rate are collected with h_1 of Figure 5 as the source node of the packet. The simulation results with 5-minute simulation time are shown in Figures 7 and 8. Observe from the figures that the RTT and packet loss rate of the proposed G-ACO scheme are almost same as the ACO scheme. When the simulation time is as small as 5 min the computation overhead of the GA in the initial phase is high while the accumulated information required for the path selection with the proposed G-ACO scheme is insufficient. This causes similar performance between G-ACO and ACO.

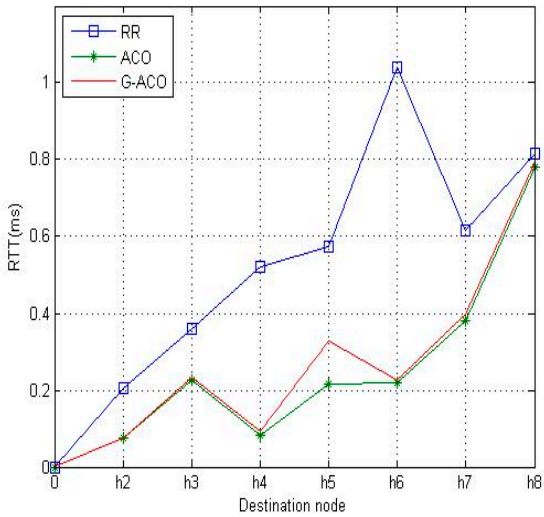

Figure 7. The comparison of RTTs with 5-minute simulation time.

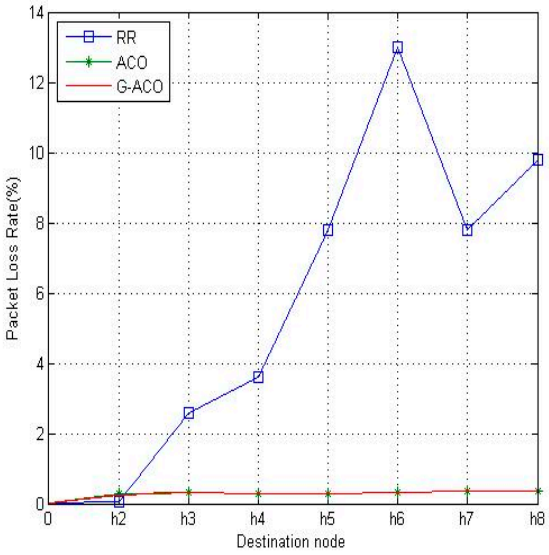

Figure 8. The comparison of packet loss rates.

Figures 9 and 10 are the results with the increased network running time of 10 min. Notice that the proposed G-ACO scheme significantly reduces the RTT and packet loss rate compared to the other two schemes. This is due to the fact that more information can be accumulated as the operation of the network lasts longer. Recall that the proposed G-ACO scheme takes advantages of the merits of both GA and ACO, the fast global search of GA and optimal search of ACO.

From the simulation results above, it was identified that the proposed G-ACO scheme substantially reduces the RTT and packet loss rate compared to the other two schemes. Notice that the packet loss rate of h_1–h_6 is very high with RR. This is because the load of h_1–h_6 causes congestion on the path. For this load, the time delay and packet loss rate are very large. With ACO, RTT is slightly improved compared to RR, while packet loss rate is significantly reduced as around 0.28% (h_2)~0.38% (h_8). This indicates the effectiveness of the use of the ACO algorithm in distributing the load. Notice that the proposed G-ACO algorithm is very effective for LB as the time delay and packet loss rate are almost the same regardless of the destinations. In particular, the packet loss rate is as low as 0.13% (h_2)~0.20% (h_8).

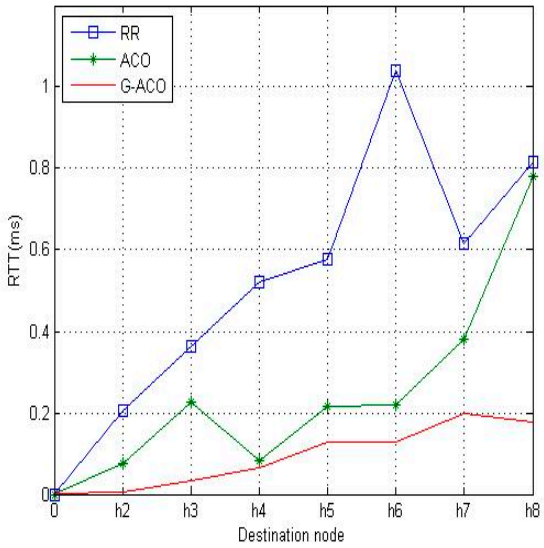

Figure 9. The comparison of RTTs with 10-minute simulation time.

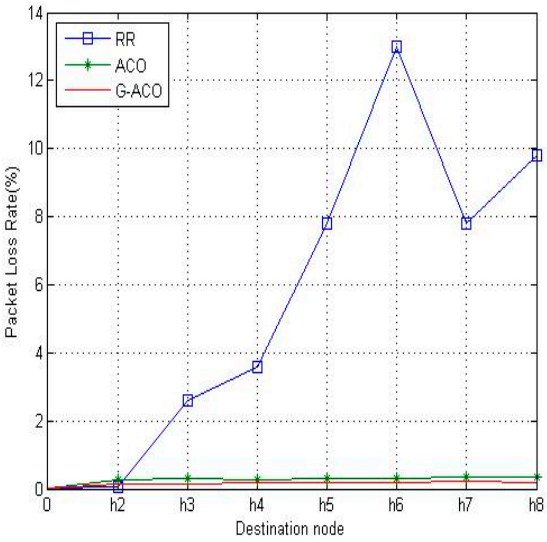

Figure 10. The comparison of packet loss rates.

Table 6 lists the path from h_1 to other hosts. Observe that same paths are taken by the RR and ACO scheme, while the proposed G-ACO scheme more widely distributes the packets throughout the network than the other schemes. With RR and ACO, seven nodes (e_7, a_3, e_8, c_2, a_6, e_9. e_{10}) are visited for the seven destinations, while nine nodes (e_7, a_4, e_8, c_1, a_6, e_9, a_3, e_9, e_{10}) are visited with the proposed G-ACO scheme. With the involvement of two additional nodes, the packets can be more evenly distributed, which allows the enhancement in the performance of networking.

Table 6. The selected path with the three schemes.

Destination	RR & ACO	G-ACO
h_2	h_1-e_7-h_2	h_1-e_7-h_2
h_3	h_1-e_7-a_3-e_8-h_3	h_1-e_7-a_4-e_8-h_3
h_4	h_1-e_7-a_3-e_8-h_4	h_1-e_7-a_4-e_8-h_4
h_5	h_1-e_7-a_3-c_2-a_6-e_9-h_5	h_1-e_7-a_4-c_1-a_6-e_9-h_5
h_6	h_1-e_7-a_3-c_2-a_6-e_9-h_6	h_1-e_7-a_4-c_1-a_6-e_9-h_6
h_7	h_1-e_7-a_3-c_2-a_6-e_{10}-h_7	h_1-e_7-a_3-c_1-a_6-e_{10}-h_7
h_8	h_1-e_7-a_3-c_2-a_6-e_{10}-h_8	h_1-e_7-a_3-c_1-a_6-e_{10}-h_8

For more comprehensive evaluation, the schemes are also tested with another topology of 14 nodes of Figure 11 [4]. Here, *srcNode* = 1 and *dstNode* = 13, and file data are transmitted at the rate of 1 Mbps. Notice from Figure 12 that the proposed G-ACO scheme consistently reduces the run time compared to the general ACO scheme.

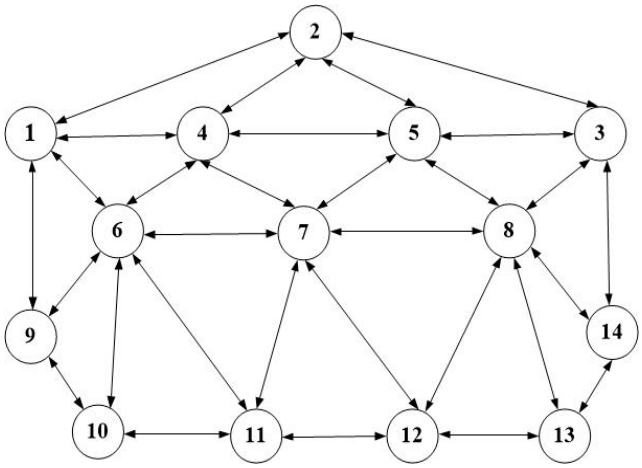

Figure 11. The 14-node topology.

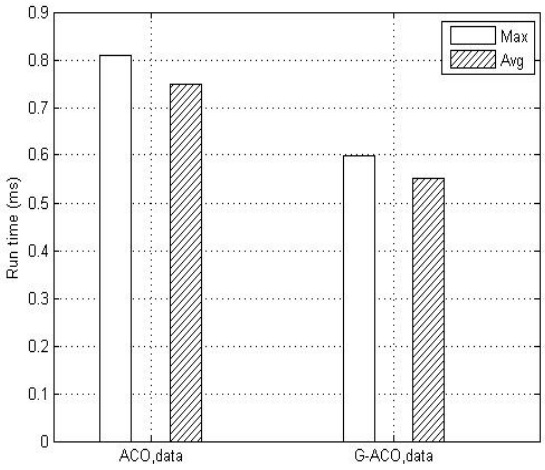

Figure 12. The comparison of running time with the 14-node topology.

5. Conclusions

In this paper we have presented the G-ACO scheme for LB of SDN. The proposed G-ACO scheme combines the selection, crossover and mutation operation of GA with the ACO algorithm to enhance the path search speed and the capability of searching an optimal path. LB is a very important issue for SDN due to the decoupling of the control plane and data forwarding plane. By implementing the proposed scheme into the LB module of OpenDayLight controller, an experiment was carried out with two networks topologies. The proposed scheme was also compared with the RR algorithm and ACO algorithm. The simulation results show that the proposed G-ACO scheme significantly improves the transmission time and packet loss rate. This is due to the effectiveness of the proposed scheme in searching the path and LB.

In the proposed G-ACO scheme, several factors were determined empirically, which significantly affect the overall performance. In the future, analytical models will be developed by which the value of the factors can be properly determined. Also, more comprehensive experiments will be conducted to investigate the relationships between the design parameters of SDN, and thereby the management of entire network can be finely tuned according to various operational conditions.

Author Contributions: Conceptualization, H.X.; Methodology, H.X. and H.Y.Y.; Software, H.X.; Validation, H.X., H.Y.Y. and K.T.K.; Formal Analysis, H.X.; Investigation, H.Y.Y.; Resources, K.T.K. and H.Y.Y.; Data Curation, H.X. and K.T.K.; Writing-Original Draft Preparation, H.X.; Writing-Review & Editing, H.Y.Y.; Visualization, H.X.; Supervision, H.Y.Y.; Project Administration, K.T.K. and H.Y.Y.; Funding Acquisition, H.Y.Y.

Funding: This work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-0-00133, Research on Edge computing via collective intelligence of hyper-connection IoT nodes), Korea, under the National Program for Excellence in SW supervised by the IITP (Institute for Information & communications Technology Promotion) (2015-0-00914), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2016R1A6A3A11931385, Research of key technologies based on software defined wireless sensor network for real time public safety service, 2017R1A2B2009095, Research on SDN-based WSN Supporting Real-time Stream Data Processing and Multi-connectivity), the second Brain Korea 21 PLUS project.

Conflicts of Interest: The authors declare no conflicts of interest.

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* **2015**, *103*, 14–76. [CrossRef]
2. Kang, S.B.; Kwon, G.I. Load Balancing Strategy of SDN Controller Based on Genetic Algorithm. *Adv. Sci. Technol. Lett.* **2016**, *129*, 219–222.
3. Li, Y.J.; Li, X.B.; Osamu, Y. Traffic Engineering Framework with Machine Learning based Meta-Layer in Software-Defined Networks. In Proceedings of the 4th IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC), Beijing, China, 19–21 September 2014.
4. Dobrijevic, O.; Santl, M.; Matijasevic, M. Ant Colony Optimization for QoE-Centric Flow Routing in Software-Defined Networks. In Proceedings of the 11th International Conference on Network and Service Management (CNSM), Barcelona, Spain, 9–13 November 2015.
5. Lin, W.C.; Zhang, L.C. The Load Balancing Research of SDN based on Ant Colony Algorithm with Job Classification. In Proceedings of the 2nd Workshop on Advanced Research and Technology in Industry Applications, Dalian, China, 14–15 May 2016.
6. Kim, H.J.; Feamster, N. Improving network management with software defined networking. *IEEE Commun. Mag.* **2013**, *51*, 114–119. [CrossRef]
7. Shin, M.K.; Nam, K.H.; Kim, H.J. Software-defined networking (SDN): A reference architecture and open APIs. In Proceedings of the 2012 International Conference on ICT Convergence (ICTC'12), Jeju Island, South Korea, 15–17 October 2012.
8. OpenFlowv1.3.0. Available online: http://www.brocade.com/content/html/en/configuration-guide/netiron-05900-sdnguide/GUID-B26EC8DB-D5A7-422E-94A0-94CC981595B3.html (accessed on 12 July 2017).
9. Zhang, H.L.; Guo, X. SDN-Based Load Balancing Strategy for Server Cluster. In Proceedings of the IEEE 3rd International Conference on Cloud Computing and Intelligence Systems (CCIS), Shenzhen, China, 27–29 November 2014.
10. Hsiao, Y.T.; Chuang, C.L.; Chien, C.C. Computer network load-balancing and routing by ant colony optimization. In Proceedings of the 12th IEEE International Conference on Networks (ICON 2004), Singapore, Singapore, 16–19 November 2004.
11. Janacik, P.; Orfanus, D.; Wilke, A. A survey of ant colony optimization-based approaches to routing in computer networks. In Proceedings of the 4th International Conference on Intelligent Systems Modeling & Simulation (ISMS), Bangkok, Thailand, 29–31 January 2013.
12. Wikipedia, Genetic Algorithm. Available online: https://en.wikipedia.org/wiki/Genetic_algorithm (accessed on 13 July 2017).
13. Sathyanarayana, S.; Moh, M. Joint Rout-Server Load Balancing in Software Defined Networks using Ant Colony Optimization. In Proceedings of the 2016 International Conference on High Performance Computing & Simulation (HPCS), Innsbruck, Austria, 18–22 July 2016.
14. Wang, Y.L.; Yuan, K.J.; Fang, W.; Liu, Y.H.; Jun, M. Research of a SDN Traffic Scheduling Technology Based on Ant Colony Algorithm. In Proceedings of the International Conference on Information Engineering and Communications Technology (IECT 2016), Shanghai, China, 21–22 May 2016.

15. Gomes, B.T.P.; Muniz, L.C.M.; Silva, F.J.S.; Santos, D.V.; Lopes, R.F.; Coutinho, L.R.; Carvalho, F.O.; Endler, M. A Middleware with Comprehensive Quality of Context Support for the Internet of Things Applications. *Sensors* **2017**, *17*, 2853. [CrossRef] [PubMed]
16. Davis, L. Applying adaptive algorithm to Epistatic Domains. In Proceedings of the International Joint Conference on Artificial Intelligence, Los Angeles, California, 18–23 August 1985.
17. Mininet. Available online: http://mininet.org/overview/ (accessed on 15 July 2017).
18. OpenDayLight Project. Available online: https://en.wikipedia.org/wiki/OpenDaylight_Project (accessed on 15 July 2017).
19. Jo, E.; Pan, D.; Liu, J.; Butler, L. A simulation and emulation study of SDN-based multipath routing for fat-tree data center networks. In Proceedings of the Simulation Conference (WSC), Savanah, GA, USA, 11–14 December 2014.
20. Charles, E. LEISERSON. Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing. *IEEE Trans. Comput.* **1985**, *34*, 892–901.
21. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **1996**, *26*, 29–41. [CrossRef] [PubMed]
22. iPerf—The TCP, UDP and SCTP Network Bandwidth Measurement Tool. Available online: https://iperf.fr (accessed on 18 July 2017).

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).