Article

Multi-Controller Load Balancing Algorithm for Test Network Based on IACO

Yanfang Fu 1, Yuting Zhu 1,*, Zijian Cao 1, Zhiqiang Du 1, Guochuang Yan 2 and Jiang Du 2

1 School of Computer Science and Engineering, Xi’an Technological University, Xi’an 710021, China; fuyanfang@xatu.edu.cn (Y.F.); caozijian@xatu.edu.cn (Z.C.); duzhiqiang@xatu.edu.cn (Z.D.)

2 The NORINCO Group Testing and Research Institute, Weinan 714200, China; yanguochuang@163.com (G.Y.); dujiang20090927@163.com (J.D.)

* Correspondence: zhuyuting@st.xatu.edu.cn

Abstract: With the rapid increase of volume and complexity in the projectile flight test business, it is becoming increasingly important to improve the quality of the service and efficiency of multi-domain cooperative networks. The key for these improvements is to solve the problem of asymmetric load of multi-controllers in multi-domain networks. However, due to the current reality, it is difficult to meet the demands of future tests, and there is not guarantee of subnet multi-domain test load balancing. Most recent works have used a heuristic approach to seek the optimal dynamic migration path, but they may fall into the local optimum. This paper proposes an improved ant colony algorithm (IACO) that can transform the modeling of the mapping relationship between the switch and the controller into a traveling salesman problem by combining the ant colony algorithm and artificial fish swarm algorithm. The IACO not only ensures the load balancing of multi-controllers but also improves the reliability of the cluster. The simulation results show that compared to other algorithms such as traditional ant colony algorithms and distributed decision mechanisms, this IACO achieves better load balancing, improves the average throughput of multi-controller clusters, and effectively reduces the response time of controller request events.

Keywords: software-defined network; multi-controller; load balancing; ant colony algorithm; artificial fish swarm algorithm

1. Introduction

Under the conditions of information technology, the trend of comprehensive joint testing is aimed at the development of the environment of multi-domain collaborative testing networks. Due to the low networking efficiency and poor configuration flexibility of current traditional networks, it is gradually becoming more difficult to meet large-scale collaborative networking tests. In response to complex and changeable test tasks, a large number of manual deployments are required to meet the requirements of these test tasks, resulting in low test efficiency and increased error rates. Therefore, using the idea of separating the control plane and the forwarding plane of SDNs and the flexibility of virtualization technology, the basic architecture scheme of SDN technology networks is selected to make the network have programmable characteristics, allow it deal with the complexity of network management and control, and enable network management and control. Additionally, the configuration is more automated and intelligent, meeting the high-reliability and real-time performance requirements of the equipment test control system network. As shown in Figure 1, the test network platform is connected with abundant and massive heterogeneous devices (high-speed camera, radar, photoelectric theodolite, telemetry equipment, etc.), and due to the inflexible and rigid network architecture, it is very difficult to achieve the centralized control and flexible configuration of these heterogeneous devices in traditional networks.
Currently, the emergence of software-defined networks (SDN) [1] provides a new scheme to solve the problem of inflexible configuration. The main feature of an SDN is the decoupling of the control plane and the data plane. It has programmable and automated features. In order to adapt to the constantly changing network environment, operators can flexibly build through programmable networks. As shown in Figure 2, SDN network architecture is divided into an infrastructure layer, a control layer, and an application layer, and the communication between these three layers is conducted through southward interface and northward interface, respectively. The infrastructure layer is composed of an SDN switch, which is responsible for data forwarding and processing. The control layer is the core layer of the architecture, which is responsible for collecting network status, calculating routing, and sending control information to the data forwarding equipment of the infrastructure layer. The top layer is the application layer, where a variety of services and different types of applications run.

**Figure 1.** Schematic diagram of multi-domain collaborative test network.

**Figure 2.** SDN communication network framework diagram.

The idea of separating the control layer and the forwarding layer in an SDN is very suitable for the construction of complex test networks. Firstly, various network scenarios in the test network are usually non-persistent, so it is an important application index...
to transform the existing network structure flexibly. Second, the test network needs to be connected to a variety of network space equipment. The connection needs to realize dynamic hybrid networking with virtual nodes, which also puts forward high requirements for the flexibility of the network configuration. In the single-domain test network scenario, due to the influence of the controller capacity, business processing capacity, computing power, and other factors, a single controller is often insufficient to control and manage the entire network behavior, so it is necessary to build a controller cluster. In this context, a multi-controller cluster based on an SDN is proposed, which mainly solves the problem of a single controller not being able to meet control and management needs and to improve the business load balancing ability in the multi-controller environment under the increasing network scale environment. In reality, the rapid increase in heterogeneous interconnected test equipment, network traffic, and business requirements has caused a load imbalance between controllers, resulting in insufficient resource utilization and reduced network performance. For this reason, the IACO algorithm is proposed to solve this problem by maximizing the load balance of the controller through switch migration. In this paper, a multi-controller cooperative system scheme and an improved ant colony algorithm are used to solve the scalability problem of the control plane and the controller overload problem under a multi-controller cluster.

Multi-controller load balancing methods can be divided into controller optimization methods and switch migration methods. The controller optimization method is used to determine the optimal number and optimal position of the controller on the network through an optimization algorithm to achieve load balancing. On the other hand, the switch migration method is used to reach the controller load balance by migrating the switch of the overloaded controller. This paper uses the switch migration method to complete the remapping of the relationship between the switch and the controller, which improves the flexibility of the network and dynamic load balancing.

The current research mainly focuses on the following three sub-points:

1. The demand for projectile flight test business and test network equipment has increased sharply, and the management scope and processing capacity of a single controller are limited, which cannot meet the real-time service requirements of various businesses.
2. In a multi-domain test network, according to the actual needs of the task, the load of the multi-controller cluster is unbalanced, which may cause the performance of the cluster to be reduced or paralyzed.
3. On the basis of satisfying multi-controller cluster load balancing, how can less migration cost be used for efficient switch migration?

The main contributions of the paper are summarized as follows:

- We develop a distributed system architecture based on multiple controllers and propose a switch dynamic migration scheme. The application of the combination of the ant colony algorithm and the artificial fish swarm algorithm in the SDN multi-controller environment is proposed, and the switch dynamic migration problem is modeled as a traveling salesman problem (TSP), and the migration target controller is obtained.
- We calculate the selection probability of the migration switch using the collected topology information. Based on the ant colony algorithm, a more reasonable interval adjustment is adopted for the volatilization factor. The concept of the crowding degree in an artificial fish school is introduced, which enhances the ability of the algorithm in optimizing the cost of the target controller selection.
- The verification method of this experiment is to create a simulation experiment that compares the IACO proposed in this article with random, ACO [2], GA-ACO [3], and DDM [4]. Through throughput, response time, load index, balanced migration times, and load indexes of different topologies, five of these indicators are verified, and the simulation results show that the IACO achieves a better balanced-load effect in a multi-controller environment.
The rest of this paper is organized as follows: The Section 2 introduces the related work. The Section 3 conducts the analysis and the modeling of related problems and describes the implementation of IACO. The Section 4 presents our simulation experiments and comparison results followed by our conclusion and future research directions in the Section 5.

2. Related Work

2.1. Research on Multi-Controller

With the increasing number of switches, it is often difficult for a single controller to control the behavior of the whole network. To overcome this problem, some researchers have proposed the implementation of a logic-centralized controller function through the joint operation of multiple controllers. Solving the problem of limited resources for a single controller has become a frequent topic in the research on SDN cluster control technology [5,6]. Though multiple controllers provide more load capacity for data streams in large-scale networks, the difference in the load capacity of each controller has a great impact on the work efficiency. For the current problems, the current multi-controller structure can be roughly divided into two categories: a horizontal structure, such as Hyperlow [7], Onix [8], and Ethane [9], or vertical structure, such as Kandoo [10]. The realization of the vertical structure superimposes a high-level control layer on multiple controllers to coordinate the communication between multiple controllers and to complete the communication request between the controllers. The resource utilization of this architecture is not high, so if the top-level node has a heavy load, the cluster will be paralyzed after the failure of the top-level node. In a horizontal structure, all of the nodes are at the same level. They have the same qualifications, and no hierarchy is involved. On the other hand, this architecture can reduce the performance requirements of a single controller and can improve the scalability of the controller.

2.2. Load Balancing Problem

According to [11,12], due to the increasing problem of the load balancing of multi-controllers, the primary solution to solve this problem is to migrate the switch. How to dynamically migrate the switch to the target controller is called the switch migration problem (SMP) [13]. In [14–16], the load balancing is determined by whether the specified threshold is reached for load balancing. Basically, load balancing occurs under three types of circumstances: (1) In [17–19], it was proposed that due to dynamic changes in network traffic, there is a need to reduce communication costs and to meet the shortage of available resources by shutting the system down and by adding some controllers; (2) in order to deploy multiple controllers to minimize the communication delay between the switch and the control plane, it is necessary to choose strategically placed controllers to optimize network performance [20–22]. (3) When a load imbalance occurs, the switch is migrated from an overloaded controller to an underloaded one. Thus, load balancing is achieved by dynamically changing the relationship between the switch and the controller [23–25].

2.3. Load Balancing Algorithms

This section reviews recent and related work using the proposed method. The load of the controller is sorted first and is then balanced by a greedy algorithm [14,19,26]. However, the results from the greedy algorithm may not be optimal, and there is a gap between the greedy algorithm and the global optimal solution. The simulated annealing algorithm was proposed to improve the load balancing efficiency [17,18]. The approach requires a large number of experiments to obtain parameters to ensure its convergence to the optimal. F. Al-Tam et al. [13] proposed a mixed-integer linear programming model to solve the SMP problem in software-defined networks, but due to the use of precise algorithms with fewer decision variables, the resulting solution may not be the optimal solution. In [27–29], a new Elasticon architecture was proposed that was based on the switch dynamic mapping architecture of multiple controllers. Through the design of a new OpenFlow protocol
and by adopting a random migration strategy (random), although the delay between the controller and the switch can be reduced, the migration is less efficient and is prone to secondary overload when the target controller is loaded. Zhong Yuan [2] proposed a load balancing strategy (ACO) based on an ant colony algorithm, applying ACO to the SDN network architecture to achieve the load balancing of the communication link, but this solution was only for a single controller and does not apply to a multi-controller environment. In [3], an algorithm that combines ant colony and genetics is proposed, which enhances the global optimization ability of the algorithm, but the parameters in the algorithm are basically static and cannot be dynamically adjusted according to the transformation of the environment. Oladipupo Adekoya et al. Hu et al. [4] proposed a distributed decision mechanism (DDM) based on switch migration in multi-subdomain SDN networks. The DDM first constructed a distributed migration decision field and then selected the migration switch and target controller according to the selection probability before the migration countdown was set to achieve orderly switch migration. However, the DDM used a greedy algorithm to select the target controller so that the results may not be the global optimal solution. Ref. [24] proposed an improved switch migration decision algorithm (ISMDA). The improved algorithm uses the controller variance and the average load state of the controller to determine the set of underloaded controllers in the network, but the migration cost is not considered to be enough. This leads to a decrease in migration efficiency. W. H. F. Aly et al. [30] proposed the most recent migration strategy. By selecting the nearest controller as the migration controller, only the migration cost between the migrated switch and the target controller is considered. However, the problem of secondary overload caused by load balancing migration was not considered. Zhang et al. [31] proposed an algorithm for solving SMP using online controller load balancing; the algorithm was used to continuously improve the load balancing degree until no switches need to be migrated. However, frequent switch migration will increase the additional overhead of the network resources and will reduce the performance of the entire network architecture. K. S. Sahoo et al. [32] proposed a controller adaptation and migration decision framework that is able to determine when to perform switch migration by monitoring the load in real time. Multi-criteria decision technology was introduced to select target controller, but the scheme monitored network status in real time and reduced efficiency without considering migration cost. In [33], an ant colony algorithm that comprehensively considers the load balancing of data link and server is proposed to select the optimal server. However, this algorithm only supports the single-controller environment, and it is difficult to meet the current demand when business increases sharply, thus leading to controller failure. In [34], an ant colony system algorithm (ACS) is proposed, which uses the server load and network statistics collected by the controller to find the best server and the best path for data flow. However, in finding the best path, the ant colony algorithm tends to converge prematurely and thus falls into the local optimal, which may not obtain the global optimal path. Reference [35] introduced an additional use of the ant colony algorithm and other algorithms in combination to avoid falling into the local optimal solution of a wide range of solutions, but we have put forward IACO as the combination of the ant colony algorithm and the artificial fish algorithm and the dynamic adjustment of the ant colony algorithm volatile factors used in SDN controller in the cluster, thus avoiding the ant colony algorithm getting trapped in the local optimal solution and optimal efficiency. Reference [36] proposed an ant colony algorithm (ACO) and a dynamic routing method to calculate the path between the source and the target using the ant colony optimization realized in the SDN. However, the variables of the ant colony algorithm in this paper are all set values, so the adaptability of the algorithm to the dynamic changes of the environment may be reduced, and the obtained may not be the optimal solution. In this paper, we further improve the efficiency of load balancing by improving the switch migration algorithm.
3. Materials and Methods

3.1. Problem Modeling

As shown in Figure 3, since projectile test equipment needs to be deployed and networked in a large range, a single controller cannot meet the unified management of a large number of heterogeneous Internet of Things devices. Therefore, the emergence of multi-controller architecture solves the problem of insufficient resources of a single controller. However, the load balancing among the controllers is the key to improving the efficiency of the controller cluster and has become a hot topic for recent research.

![Figure 3. Schematic diagram of multi-domain collaborative test networking.](image)

In this paper, we remap the relationship between the switch and the controller through dynamic switch transfer. In multi-controller SDN, the controller has three identities: equal, master, and slave. The master controller is the only controller in the domain that can update the switch flow table. In other words, each switch cannot have more than one master controller, but there can be one or more equal slave controllers [37]. When the master controller fails, an equivalent controller or a slave controller can replace it. The task of changing the master controller of a switch is called switch migration.

As shown in Figure 4, the data layer is represented by the controller connected to each device in the launch zone, navigation zone, and drop zone in Figure 3. The controller layer is the controllers C1, C2, and C3 of the three zones. The devices are all connected to switches, so they are simulated as seven switches managed by controllers in three zones. As the Figure shows, the switch S5 is connected to the controller C1. C1 is the master controller of S5, and the remaining controllers are the slave controllers of S5. When the load of the main controller C1 suddenly increases above the set load threshold at a certain moment, switch migration is required, and the lighter-loaded C3 is selected from the remaining slave controllers for migration. This paper studies the dynamic migration of the selected switch and controller after determining the migrated switch using IACO.

![Figure 4. Schematic of SDN under multi-controller.](image)

We set all of the controller nodes in the SDN area as $C = \{C_1, C_2, C_3, \ldots, C_M\}$, all of the switches as $S = \{S_1, S_2, S_3, \ldots, S_M\}$, and load balancing conditions. When the controller
is overloaded, the scheme obtained by the improved algorithm is used for migration to achieve load balancing.

**Definition 1.** Switch distribution matrix \( F = (f_{ij})_{M \times N} \), which indicates the mapping relationship between the controller and the switch, \( f_{ij} = 1 \), meaning \( c_i \) control \( s_j \). \( f_{ij} = 0 \) means that switch \( s_j \) is not controlled by controller \( c_i \).

**Definition 2.** Packet-in load \( \lambda_i \) of controller \( C_i \), which refers to the sum of the packet-in message rates of each switch managed:

\[
\lambda_i = \sum_{j \in s_i} \lambda_i,j
\]  

(1)

**Definition 3.** Packet-in load index \( L_i \) of controller \( C_i \), which represents the ratio of the current packet-in load to its maximum processing capacity \( \lambda_{\text{max}} \):

\[
L_i = \frac{\lambda_i}{\lambda_{\text{max}}}
\]  

(2)

**Definition 4.** The average packet-in load index \( L_{\text{avg}} \) of controllers in all controller clusters \( i \):

\[
L_{\text{avg}} = \frac{1}{|n|} \cdot \sum_{i \in C} L_i
\]  

(3)

**Definition 5.** Define the packet-in of each controller in the cluster, and the load balance degree is expressed as the standard deviation of the packet-in load of the controller cluster as:

\[
\sigma = \sqrt{\frac{1}{|n|} \sum_{i \in C} (L_i - L_{\text{avg}})^2}
\]  

(4)

Therefore, the optimized objective function can be obtained as:

\[
T_{\text{best}} = IACO_{\text{best}}
\]  

(5)

\[
L_i < 0.8 \quad \forall i
\]  

(6)

\[
f_{ij} \in \{0, 1\} \quad \forall i, j
\]  

(7)

\[
\sum_{i \in G} f_{ij} = 1 \quad \forall j
\]  

(8)

The objective function Formula (5) is the optimization goal of the model, and \( T_{\text{best}} \) is the optimal path of the overload controller migration obtained by the improved ant colony algorithm. The objective function Equations (6)–(8) are the model constraint functions. Function (6) ensures that the load index of all of the controllers is lower than 0.8 to avoid controller failure, so it is also a trigger condition for load balancing. Function (7) represents the connection status of each network device. Function (8) ensures that the OpenFlow switch in the entire cluster \( G \) has only one master controller.

In this paper, the switch migration problem is abstracted as the traveling salesman problem (TSP), a complex graph close to the TSP problem is obtained, and we attempt to solve the optimization problem and obtain the optimal migration path.

### 3.2. IACO

To balance the controller load at a small cost, we proposed an improved ant colony algorithm (IACO) for dynamic migration. Below are the details of the IACO’s two phases of realization: the selection of the migration switch and the obtaining of the optimal migration controller.
3.2.1. Switch Selection

If a controller $C_i$ is overloaded, then the $C_i$ may select a controllable changeover switch $S_j$ based on a specific probability distribution. The trigger of this selection is controlled by two factors: the number of flow requests and the delay of the controller and the switch. For the former, because each migration of the switch will affect the system, the switch with less flow requests will be migrated by the controller first. For the latter, since the switch management cost is high, there is a higher probability of selection. Therefore, the probability formula $P_{ij}$ is used here to select the switch.

$$P_{ij} = \frac{\exp(-r_{ij}t_{ij})}{\sum_{j \in S_i} \exp(-r_{ij}t_{ij})}$$

(9)

where $r_{ij}$ is the number of flow requests of the switch, $t_{ij}$ is the delay between $C_i$ and $S_j$, and $S_i$ is the set of $S_j$ controlled by $C_i$.

3.2.2. Target Controller Selection

After the relocation controller and the relocation switch in the domain of the controller are decided, we combine the ant colony algorithm and artificial fish swarm algorithm to obtain the optimal migration controller. Although the ant colony algorithm uses the principle of pheromone positive feedback to find the best path, it is easy to fall into the optimal solution locally. Hence, we introduce the definition of the crowding degree of the artificial fish swarm to prevent the ant colony algorithm from falling into the local optimal and prematurely mature phenomenon so that the ability of the approach to find the global optimal is improved. For the ant colony algorithm to solve the controller selection problem under the multi-controller architecture, the most commonly used method is to convert the problem into the classic traveling salesman problem (TSP), obtain a complex graph similar to the TSP problem, and then solve its optimization problem using the IACO, which uses the ant pheromone concentration to represent the migration path. When the conditions of the algorithm are satisfied, the more path pheromones that are released, the greater the probability of the path becoming the best migration path is.

The congestion degree of artificial fish swarm algorithm is defined as

$$q_{ij} = \frac{2\tau_{ij}(t)}{\sum_{i \neq j} \tau_{ij}(t)}$$

(10)

where $\tau_{ij}(t)$ represents the concentration on the path in the ant colony algorithm. Define the congestion threshold at the time $t$:

$$\gamma(t) = 1 - \frac{1}{e^{ac}}$$

(11)

where $a$ is the threshold coefficient, and $c$ is the number of cycles.

If $q_{ij} < \gamma(t)$, the current path is unobstructed, and the ant will continue moving along the current path without reselecting the next feasible path. Otherwise, the ant will randomly choose a feasible path to move around the neighborhood.

An ant colony algorithm is a biological bionic algorithm that uses ant behavior to transmit information through pheromones to establish an ant colony algorithm model around those pheromones. The following formula calculates the probability that the ant $k$ chooses the next city $j$ when it is in the city $i$. Where $allowSet_k$ is the next set of cities that ant $k$ can choose at time $t$, $\tau_{ij}(t)$ represents the pheromone on the path between city $i$ and city $j$ at the time $t$, and $a$ is the dependence of pheromone $\tau_{ij}(t)$, and where $\eta_{ij}$ is the heuristic
factor, which is generally the reciprocal of the pheromone concentration $\tau_{ij}(t)$ and $\beta$ is the dependence of the heuristic $\eta_{ij}(t)$:

$$P_{ij}^k(t) = \begin{cases} 
\frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^\beta}{\sum_{j \in \text{allowSet}_k} [\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^\beta} & \text{if } j \in \text{allowSet}_k \\
0 & \text{otherwise}
\end{cases} \quad (12)$$

Define the number of bytes transmitted at $t_1$ to $t_2$ each time period

$$B_i(t) = \frac{\text{Bytes}(t_1 - t_2)}{t_1 - t_2} \quad (13)$$

$$\tau_{ij}(t) = \frac{B_i(t)}{B_{\text{max}}^i(t)} \times 100\% \quad (14)$$

$$\eta_{ij}(t) = \frac{1}{\tau_{ij}(t)} \quad (15)$$

Our IACO approach improves the concentration update method in two aspects: the local update and the global update. The local update is that for the ant $k$, given the path $<i, j>$ that it travels, the pheromone update is defined as:

$$\tau_{ij}(t + 1) = (1 - \rho)\tau_{ij}(t) + \rho\tau_{ij}(0) \quad (16)$$

$\rho$ is the volatile factor, and $\tau_{ij}(0)$ is the initial value of the pheromone:

$$\tau_{ij}(0) = L_i \cdot \alpha + \sigma \cdot \beta \quad (17)$$

where $\tau_{ij}(0)$ is the initial target value of IACO, $L_i$ is the load index of the controller, and $\sigma$ is the packet-in load standard deviation of each controller in the cluster. The global update $\chi$ is to update the pheromone on the current path after the ant completes the iteration. It is defined as

$$\chi \begin{cases} 
\tau_{ij}(t + 1) = (1 - \rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t) \\
\Delta\tau_{ij}(t) = \frac{L_C - L_{\text{best}}}{L_C} \quad (18)
\end{cases}$$

where $\Delta\tau_{ij}(t)$ is the global pheromone increment, $L_C$ is the optimal path length of the current iteration that is the $C$-th iteration, and $L_{\text{best}}$ is the current optimal path.

According to [38], the IACO adjusts the pheromone volatilization factor $\rho$ between partitions as

$$\rho = \begin{cases} 
0.9, & \text{if } C \in [0, 0.2 \cdot C_{\text{MAX}}) \\
0.7, & \text{if } C \in [0.2 \cdot C_{\text{MAX}}, 0.4 \cdot C_{\text{MAX}}) \\
0.5, & \text{if } C \in [0.4 \cdot C_{\text{MAX}}, 0.6 \cdot C_{\text{MAX}}) \\
0.3, & \text{if } C \in [0.6 \cdot C_{\text{MAX}}, 0.8 \cdot C_{\text{MAX}}) \\
0.1, & \text{if } C \in [0.8 \cdot C_{\text{MAX}}, C_{\text{MAX}}) 
\end{cases} \quad (19)$$

where $C_{\text{MAX}}$ represents the maximum number of cycles set.

Based on the discussion above, we summarize the IACO in Algorithm 1. It has two stages. One is to find the best solution of the migration controller, and the other is to complete the redeployment of the relationship between the switch and the controller.

More specifically, in the first stage, we first initialize the pheromone moment that is configured by the distance matrix and the greedy algorithm, and some parameters, including pheromone importance factor $\alpha$, heuristic factor $\beta$, the congestion threshold coefficient $\mu$, the congestion degree $\sigma(t)$, the total number of iterations $c$, and the number of ants $m$ and the number of nodes $n$ (steps 1–4). Next, it needs to be determined whether it is less than the iteration period $c$, and if it is satisfied, then it is appropriate to proceed to the next step, otherwise the loop traversal should end (steps 5). After all of the ants go through a path node (steps 6–8), the ant’s next city and the congestion degree $\sigma(t)$ to reach the next city need to be calculated through the roulette algorithm and then whether it is less
than the threshold $\gamma(t)$ needs to be determined (steps 9–12). After the ants have walked all the paths, the list of cities that the ants can reach is updated (steps 13–14). Subsequently, the path length of each ant is calculated, the appropriate volatilization factor according to the number of iterations is selected, and the information of the entire topology element is updated (steps 17–29). In the second stage, the selected migration switch is migrated to the target controller obtained in the first stage, provided that the preset load balance is satisfied.

**Algorithm 1: IACO**

**Stage 1 Target controller selection**

**Input** $G = (V,E)$  
**Output** $C_{objective}$

1) for each edge  
2) set initial pheromone value  
3) end for  
4) set value $\alpha$, $\beta$, $\rho$, $\mu$, $\sigma$, $c$, $m$  
5) while $t < c$  
6) for each ant $k$  
7) randomly choose an initial city  
8) for $i=1$ to $n$  
9) if $\sigma(t) < \gamma(t)$  
10) choose next city $j$ with probability  
11) else  
12) randomly choose another city  
13) end if  
14) update list of allowed city of ant $k$  
15) end for  
16) end for  
17) compute the length $C_k$ of the tour constructed by the $k$th ant  
18) for each edge  
19) update the pheromone value  
20) end for  
21) end while  
22) print result $C_{objective}$

**Stage 2 Switch dynamic migration**

**Input** Target $C_{objective}$  
**Output** New mapping relationship after completing the migration

23) execute $S_{migration}$ to $C_{objective}$

The complexity analysis of IACO: for the steps 1–3, the time complexity of $M$ cycles is $O(M)$; for the steps 7–17 when the number of ants $K$ is small, the time complexity is $O(K*N)$ or $O(N^2)$; otherwise, the time complexity of the $M$ traversals in steps 19–22 is $O(M)$. Hence, the total time complexity of the final algorithm is $O(N^2)$.

4. Experimental Simulation and Analysis

4.1. Simulation Environment

To study the ability of the IACO in terms of load balancing, throughput, response time, and communication overhead in the multi-domain test network, we built the multi-domain collaborative test network diagram shown in Figure 3 and used Mininet software [39] to build the simulation topology environment shown in Figure 4. Three floodlight controllers and seven OpenFlow switches were used to build a multi-controller cluster, and the simulation experiment was conducted under the Ubuntu16.04 system. To verify that the IACO can be deployed in different topological environments with good load balancing ability, a comparison with the other two different topological environments is made to
prove that the load balancing ability of the IACO is suitable for a variety of different network topological environments.

4.2. Experimental Parameter Settings

Combining the actual situation and characteristics of the dynamic migration algorithm in the SDN multi-controller architecture, the conditions for triggering load balancing are set as follows: load index greater than or equal to 0.8, sampling cycle $T = 1s$, and crowding threshold $c = 0.003$. The request rate of the switch should be set to be within the dynamic range, and the average flow rate should be set to 512 kbit/s. At the same time, in order to reflect the difference of the controller status in each sub-domain, the controller processing capacity should be 12 MB~15 MB. The values of the heuristic factors $\alpha$ and $\beta$ and the number of ants $m$ in the IACO are obtained by simulation experiments on the multi-controller dynamic migration architecture.

Because the value of the pheromone volatilization coefficient $\rho$ is regulated by (19), it is only necessary to conduct simulation experiments with different combinations of heuristic factors $\alpha$ and $\beta$. The simulation results of the influence of the values of heuristic factors $\alpha$ and $\beta$ on the IACO are shown in Table 1. The “iterations” in the table represent the number of loops of the final solution obtained by the algorithm. We need to refer to the influence of these two influence factors on the algorithm to find a balance point.

Table 1. Influence of heuristic factors $\alpha$ and $\beta$.

| $\alpha$ | $\beta$ | Iterations |
|---------|--------|------------|
| 0.1     | 0.1    | 35         |
| 0.1     | 0.5    | 23         |
| 0.5     | 1      | 14         |
| 1       | 2      | 9          |
| 3       | 8      | 4          |
| 6       | 9      | 2          |

When $\alpha$ and $\beta$ are both large, $\alpha$ makes the ant pay too much attention to the pheromone concentration when searching for the path, and even ACO, which completely relies on the pheromone concentration to find the path, will converge too quickly. However, when the network topology is large, it is easy to fall into a local optimal solution. When $\alpha$ and $\beta$ are both small, $\alpha$ basically makes the ants ignore the pheromone concentration in the process of finding a path, and the ants do not even rely on the pheromone concentrations to make choices. Therefore, we eventually chose $\alpha = 1$ and $\beta = 2$.

As shown in Table 2, the number of ants for the IACO also affects the overall performance of the algorithm. If the number of ants is too small, so the solution that is set is too small, so the search ability of the whole ant network is weakened, and it is easy to fall into the local optimal solution. When the number of ants increases, the solution set is to be larger, which will enhance the randomness of the global search. This also slows down the convergence of the algorithm. Therefore, we eventually chose the number of ants to be $m = 8$.

4.3. Performance Evaluation and Testing

In a SDN network, the load balancing performance of the controller mainly indicates the robustness of the whole cluster when the load of a controller is heavy. The performance of the controller can be verified by three indicators: throughput, controller load rate, and packet-in response time before and after migration.

The simulation results show that the proposed algorithm can provide a faster and more reliable balanced-load effect. In order to verify the performance of the IACO, a series
of simulation experiments were established, and IACO was compared to random, ACO, GA-ACO, and DDM.

**Table 2. Influence of the number of ants.**

| Ant Number | Iterations |
|------------|------------|
| 3          | 21         |
| 6          | 13         |
| 8          | 9          |
| 12         | 6          |
| 15         | 2          |

4.3.1. Throughput

Network throughput refers to the number of requests processed by the server in a unit time, which reflects the bearing capacity of the server and is an important tradeoff benchmark for the processing capacity of the system. We evaluated the system throughput by continuously increasing the connection request of the system.

In this experiment, as shown in Figure 5, we can see that there is little difference in the throughput of the five algorithms before 31s. After 31s, when we increase the packet-in message sending rate of the switch, it is obvious that the throughput of random cannot complete the real-time processing of the packet-in messages. Because of the increase in the number of requests, the load of the controllers will be unbalanced. As a result, some controllers will be overloaded, and the requests cannot be processed, while other controllers are still under low load, resulting in a decrease in the overall throughput of the controller cluster. The throughput of ACO, GA-ACO, and DDM is lower than that of IACO. Overall, the throughput of the IACO is increased by about 19%, 11.8%, 8.8%, and 5.6% more than Random, ACO, GA-ACO, and DDM, respectively.

![Figure 5. Comparison of throughput after load balancing.](image)

4.3.2. Controller Load Index

The controller load index after load balancing is an important benchmark for measuring performance. We set the load index to exceed 0.8 as the condition to start the balanced load. The size of the controller load index will affect the switch migration ability. After achieving load balancing, the lower and more average load index of the controller can reflect the load balancing degree of the controller well. Figure 6 compares the results on the load index.
In SDN network, the load balancing performance of the controller mainly indicates the robustness of the whole network service quality for the entire cluster. The performance of the controller can be verified by three indicators: network throughput, controller load rate and packet-in response time before and after migration.

4.3.1 Throughput
Throughput is an important indicator to evaluate the system throughput. We evaluate the system throughput by continuously increasing the connection request of the system. In the experiment, the topology model OS3E, Topology zoo and custom topology were used to verify the topology adaptability of the algorithm. The network throughput refers to the number of requests processed by the server in a unit time, which reflects the well reflect the load balancing degree of the controller. Figure 6 compares the results on the load index.

Figure 6. Comparison of load index before and after load balancing.

It can be seen from Figure 6 that apart from random, the other algorithms can complete the migration of the switches in the overload controller so as to achieve the effect of load balancing. The standard deviation results of the load index of random, ACO, GA-ACO, DDM, and IACO are 0.03207, 0.01129, 0.00542, 0.00329, and 0.00047, respectively. Obviously, random has the worst ability to achieve load exponential balance. The reason for this could be that although random reduces the load index to a certain extent; it cannot achieve load balance well and there is a possibility of secondary overload. Other algorithms prevent the occurrence of secondary overload. In fact, the IACO is the best among all of the algorithms in terms of load index and the load balance.

4.3.2 Controller load index
The controller load index after load balancing is an important benchmark for measuring performance. We set the threshold of the load index to exceed 0.8 as the condition to start the balanced load. The size of a controller load index will affect the throughput of the controller cluster. The throughput of ACO, GA-ACO, and DDM is lower than that of IACO. Overall, the throughput of IACO is increased by about 19%, 11.8%, 8.8%, 5.6% than Random, ACO, GA-ACO, DDM, respectively. After achieving load balancing, the lower and more average load index of the controller can also be that although random reduces the load index to a certain extent; it cannot achieve load balance well and there is a possibility of secondary overload. Other algorithms prevent the occurrence of secondary overload. In fact, the IACO is the best among all of the algorithms in terms of load index and the load balance.

4.3.3 Packet-in Response Time before and after Controller Migration
The response time of the controller packet-in is another performance criterion that can be used to weigh the controller load strategy. The response time of the controller has a strong influence on the acceptance and delivery rate of the flow meter. The response time of the controller is an indicator that can be used to reflect the status of the controller in the cluster. The lower the response time is, the better the controller status is. Therefore, we conducted experiments on the response time of the packet-in events.

The experimental simulation results are shown in Figures 7 and 8. As it can be seen from the figures, at the beginning of the packet-in event, the responses of all of the algorithms are relatively close, and when the request rate increases, the response time becomes larger, which is in line with expectations. When the request rate of packet-in is low, the response time of all of the algorithms is basically the same. However, when the packet-in event rate and the load on each controller increase, the advantage of the IACO is shown. Under the worst scenario, the response time of random is more than three times that of the IACO and is about 16.7% lower than that of DDM. Figure 8 shows the packet-in response time of the different controllers of the five algorithms after load balancing. The standard deviation results of the packet-in response time of the five algorithms of random, ACO, GA-ACO, DDM, and IACO are as follows: 4.9261, 4.1096, 3.2998, 1.2472, and 0.6667. Obviously, the IACO’s response time is not only lower than other four algorithms, but it is also more balanced. This shows that the IACO has a stronger load balancing performance while ensuring a low response time, which is enough to show the advantage of the IACO in the request rate of a large number of packet-in events and that it can provide reliable network service quality for the entire cluster.

4.3.4 Different Topologies Balance the Load Index Contrast
In the experiment, the topology model OS3E, Topology zoo and custom topology were used for comparative experiments. Under different topological environments, the number of controllers and the number of switches were different, and the balance degree of the load index was used to verify the topology adaptability of the algorithm. The network topology and parameter settings are shown in Table 3.
The response time of the controller is an indicator used to reflect the status of the controller in the cluster. The lower the response time, the better the controller status is. Therefore, we conduct experiments on the response time of packet-in events to evaluate the performance of different algorithms.

The response time of the controller has a strong influence on the acceptance and delivery rate of the flow meter. Therefore, we conduct experiments on the response time of packet-in events to evaluate the performance of different algorithms. The response time of the controller is an indicator used to reflect the status of the controller in the cluster. The lower the response time, the better the controller status is. Therefore, we conduct experiments on the response time of packet-in events to evaluate the performance of different algorithms.

The response time of the five algorithms of Random, ACO, GA-ACO, DDM, and IACO are as follows: 4.9261, 4.1096, 3.2998, 1.2472, and 0.6667 respectively. Obviously, IACO's response time is not only lower than other four algorithms, but also more than three times that of IACO and about 16.7% lower than that of DDM. Figure 8 shows the packet-in response time of each controller.

### Table 3. Topology Information Table.

| Topological Name | Number of Nodes | Number of Links | Number of Controllers |
|------------------|-----------------|----------------|-----------------------|
| Customize        | 7               | 7              | 3                     |
| Topology zoo     | 18              | 29             | 4                     |
| OS3E             | 9               | 9              | 5                     |

Figures 6, 9 and 10 show the controller load balancing rate of the three schemes under different network topologies. It can be seen from Figure 9 that compared to other algorithms, the IACO’s controller load is more balanced. Because of its static deployment, random does not adapt to dynamic changes in network traffic, and its load balancing performance is the worst. ACO is essentially an ant colony algorithm, which makes it difficult for it to achieve global optimization, and it also has limited load balancing performance. DDM migrates switches based on a greedy algorithm and cannot guarantee a global optimal solution. Although GA-ACO improves the load balancing ability, the parameters in the algorithm cannot be dynamically adjusted according to environmental changes. Compared to GA-ACO and DDM, the IACO uses a hybrid algorithm to obtain better solutions, can better adapt to traffic characteristics, and achieves a more reasonable switch migration mechanism and a better load balancing rate.

#### 4.3.5. Switch Migration Times

Switch migration is the main method to solve the load balancing problem under multiple controllers, but switch migration that is too frequent may affect the quality of the entire network service. In the current network environment, each switch is managed by the main controller, and the information exchange and migration time of each switch to another controller are essentially the same and can be expressed as Com_m. The number...
of migrations to complete the entire load balancing is expressed as \( \text{Num}_m \), so we can express the communication overhead as \( \text{Com}_m \times \text{Num}_m \). Since \( \text{Com}_m \) is essentially the same, we denote the number of switch migrations as the communication overhead in load balancing.

![Figure 9. Comparison of load balance under topology zoo.](image)

![Figure 10. Comparison of load balance under OS3E topology.](image)

As shown in Figure 11, under the five load balancing methods, IACO has the least number of migrations and communication overhead after load balancing. It can be seen from Figure 11 that due to the rigidity of the random algorithm’s switch migration selection, ACO migrates switches based on ant colony algorithm, which makes it difficult for it to achieve global optimal control. Compared to GA-ACO and DDM, the IACO uses the algorithm to dynamically adjust parameters and is able to obtain a better global optimal solution through the combination of the artificial fish school algorithm. The optimized mapping of the controller-switch is realized, the number of switch migrations is reduced, and the communication overhead in the network is reduced.

![Figure 11. Number of switches migrated.](image)
5. Conclusions

Due to the rapid growth of business requirements, there may be asymmetries in the demand for controllers. This asymmetry may cause an asymmetry in the load between the controllers, which may result in a decrease in controller performance. This paper presents a new solution to the switch migration problem (SMP) in the multi-controller environment of an SDN. An improved ant colony algorithm (IACO) is proposed to address the controller’s overloading issue. The IACO improved the traditional ant colony algorithm, divided the pheromone update into the global update and the local update, and dynamically adjusted the volatile factors in the different segments. Combined with the artificial fish swarm algorithm, the concept of congestion was introduced to improve the global optimization ability of the algorithm. Simulation experiments showed that compared to other algorithms such as random, ACO, GA-ACO, and DDM, the IACO achieves a better load balancing effect and has a good load balancing degree. It improves the cluster throughput of the controller and effectively reduces the response time of request events. As such, we put forward the methods for a single domain using the controller in a scene test network capacity, with business processing capacity factors, put forward many controller schemes-balanced loads, tested the algorithm through various applications to improve multi-domain IACO cluster network reliability and availability, and thus improved the testing defense capabilities of the network platform to provide technical support.

However, the algorithm currently being studied in this paper should only be used when the current controller cluster resources meet the current demand, and it does not consider the failure analysis and the current shortage of resources when the cluster demand is met. Therefore, in the future, according to [40], the use of intelligent algorithms combined with heterogeneous Internet of Things devices based on the situational awareness of multi-domain test network environment will be an important topic for research. Additionally, load balancing by dynamically adding controllers will become the focus of future research.

Author Contributions: Conceptualization, Y.F. and Y.Z.; funding acquisition, Z.D.; investigation, Y.F., Y.Z., Z.D., and G.Y.; methodology, Y.Z.; resources, Z.C.; software, Y.Z.; validation, G.Y. and J.D.; writing—original draft, Y.Z.; writing—review and editing, Y.F. All authors have read and agreed to the published version of the manuscript.

Funding: Shaanxi Provincial Science and Technology Department: Shannxi S&T Grant 2021KW-07.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the editors and the anonymous reviewers for their valuable comments on the content and the presentation of this article.

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

References
1. Hu, Y.; Wang, W.; Gong, X.; Que, X.; Cheng, S. BalanceFlow: Controller load balancing for OpenFlow networks. In Proceedings of the IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, Hang Zhou, China, 30 October–1 November 2012; pp. 780–785. [CrossRef]
2. Zhong, Y. Research on SDN Load Balancing Technology Based on Ant Colony Algorithm. Appl. Microcomput. 2019, 35, 65–67.
3. Hongyun, C. Research on Transfer Optimization of Load Balancing in SDN Control Plane. Master’s Thesis, Xi’an Technological University, Xi’an, China, 2019.
4. Hu, T.; Yi, P.; Zhang, J.; Lan, J. A distributed decision mechanism for controller load balancing based on switch migration in SDN. China Commun. 2018, 15, 129–142. [CrossRef]
5. Xie, R.; Umair, Z.; Jia, X. A wireless solution for SDN (software defined networking) in data center networks. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6.
6. Al-Jaroodi, J.; Mohamed, N.; Jiang, H. Swanson Middleware infrastructure for parallel and distributed programming models in heterogeneous systems. IEEE Trans. Parallel Distrib. Syst. 2016, 14, 1100–1111. [CrossRef]
7. Tootoonchian, A.; Ganjali, Y. Hyper Flow: A distributed control plane for Open Flow. In Proceedings of the Internet Network Management Conference on Research on Enterprise Networking, San Jose, CA, USA, 27 April 2010; USENIX Association: Berkeley, CA, USA, 2010; pp. 3–6.
8. Koponen, T.; Casado, M.; Gude, N.; Stribling, J.; Poutievski, L.; Zhu, M.; Ramanathan, R.; Iwata, Y.; Inoue, H.; Hama, T.; et al. Onixa distributed control platform for large-scale production networks. In Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, Vancouver, BC, Canada, 4–6 October 2010; USENIX Association: Berkeley, CA, USA, 2010.
9. Casado, M.; Freedman, M.J.; Pettit, J.; Luo, J.; McKeown, N.; Shenker, S. Ethan: Taking control of the enterprise. ACM SIGCOMM Comput. Commun. Rev. 2007, 37, 1–12. [CrossRef]
10. Yeganeh, S.H.; Ganjali, Y. Kandoo: A framework for efficient and scalable offloading of control applications. In Proceedings of the 1st ACM Workshop on Hot Topics in Software Defined Networks, Helsinki, Finland, 13 August 2012; ACM Press: New York, NY, USA, 2012; pp. 19–24.
11. Hu, T.; Guo, Z.; Yi, P.; Baker, T.; Lan, J. Multi-controller based software-defined networking: A survey. IEEE Access 2018, 6, 15980–15996. [CrossRef]
12. Zhang, Y.; Cui, L.; Wang, W.; Zhang, Y. A Survey on Software Defined Networking with Multiple Controllers. J. Netw. Comput. Appl. 2017, 103, 101–118. [CrossRef]
13. Al-Tam, F.; Ashrafi, M.; Correia, N. On controllers’ utilization in software-defined networking by switch migration. In Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Proceedings of the 9th International EAI Conference, Faro, Portugal, 19–20 September 2018; Springer: Cham, Switzerland, 2018; Volume 263. [CrossRef]
14. Wang, C.; Hu, B.; Chen, S.; Li, D.; Liu, B. A switch migration-based decision-making scheme for balancing load in SDN. IEEE Access 2017, 5, 4537–4544. [CrossRef]
15. Lin, S.-C.; Wang, P.; Luo, M. Control traffic balancing in software defined networks. Comput. Net. 2016, 106, 260–271. [CrossRef]
16. Gao, X.; Kong, L.; Li, W.; Liang, W.; Chen, Y.; Chen, G. Traffic load balancing schemes for devolved controllers in mega data centers. IEEE Trans. Parallel Distrib. Syst. 2017, 28, 572–585.
17. Ye, X.; Cheng, G.; Luo, X. Maximizing SDN control resource utilization via switch migration. Comput. Net. 2017, 126, 69–80. [CrossRef]
18. Yao, L.; Hong, P.; Zhang, W.; Li, J.; Ni, D. Controller placement and flow based dynamic management problem towards SDN. In Proceedings of the IEEE International Conference on Communication Workshop (ICCW), London, UK, 8–12 June 2015; pp. 363–368.
19. Chen, Y.; Yang, Y.; Zou, X.; Li, Q.; Jiang, Y. Adaptive distributed software defined networking. Comput. Commun. 2017, 102, 120–129. [CrossRef]
20. Yao, G.; Bi, J.; Li, Y.; Guo, L. On the Capacitated Controller Placement Problem in Software Defined Networks. IEEE Commun. Lett. 2014, 18, 1339–1342. [CrossRef]
21. Zhang, C.; Wang, X.; Li, F.; Huang, M.; He, Q. Network service chains deployment across multiple SDN domains. Int. J. Commun. Syst. 2018, 31, e3826. [CrossRef]
22. Kumari, A.; Sairam, A.S. Controller placement problem in software-defined networking: A survey. Networks 2021, 195–223. [CrossRef]
23. Bang, Z.; Wang, X.; Huang, M. Dynamic controller assignment problem in software-defined networks. Eur. Trans. Telecommun. 2018, 29, 81–98.
24. Adekoya, O.; Aneiba, A.; Patwary, M. An Improved Switch Migration Decision Algorithm for SDN Load Balancing. IEEE Open J. Commun. Soc. 2020, 1, 1602–1613. [CrossRef]
25. Al-quaara, R.; Alma’aith, A. A Secure Switch Migration Scheduling Based on Prediction for Load Balancing in SDN. In Proceedings of the 12th International Conference on Information and Communication Systems (ICICS), Valencia, Spain, 24–26 May 2021; pp. 364–370. [CrossRef]
26. Lan, W.; Li, F.; Liu, X.; Qiu, Y. A dynamic load balancing mechanism for distributed controllers in software-defined networking. In Proceedings of the 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Changsha, China, 10–11 February 2018; pp. 259–262.
27. Anand, K.; Shoban, P.; Gember-Jacobson, A. Pratyaastha: An efficient elastic distributed SDN control plane. In Proceedings of the 3rd Workshop Hot Topics Software Defined Network (HotSDN), Chicago, IL, USA, 22 August 2014; ACM: New York, NY, USA, 2014; pp. 133–138.
28. Al-Tam, F.; Ashrafi, M.; Correia, N. On Controllers’ Utilization in Software-Defined Networking by Switch Migration. In Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Proceedings of the 9th International EAI Conference, Faro, Portugal, 19–20 September 2018; Springer: Cham, Switzerland, 2018; Volume 263. [CrossRef]
29. Dixit, A.; Hao, F.; Mukherjee, S.; Lakshman, T.V.; Kompella, R. Towards an elastic distributed SDN controller. In Proceedings of the 2nd ACM SIGCOMM Workshop Hot Topics Software Defined Networking, Marina del Rey, CA, US, 20–21 October 2013; pp. 7–12.
30. Aly, W.H.E.; Alanazi, A.M.A. Enhanced controller fault tolerant (ECFT) model for software defined networking. In Proceedings of the 5th International Conference on Software Defined Systems (SDS), Barcelona, Spain, 23–26 April 2018; pp. 217–222.
31. Zhang, S.; Lan, J.; Sun, P.; Jiang, Y. Online load balancing for distributed control plane in software-defined data center network. *IEEE Access* 2018, 6, 18184–18191. [CrossRef]

32. Sahoo, K.S.; Puthal, D.; Tiwary, M.; Usman, M.; Sahoo, B.; Wen, Z.; Sahoo, B.P.; Ranjan, R. ESMLB: Efficient Switch Migration-Based Load Balancing for Multicontroller SDN in IoT. *IEEE Internet Things J.* 2020, 7, 5852–5860. [CrossRef]

33. Li, J.; Yang, L.; Wang, J.; Yang, S. Research on SDN Load Balancing based on Ant Colony Optimization Algorithm. In Proceedings of the IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOECC), Chongqing, China, 14–16 September 2018; pp. 979–982. [CrossRef]

34. Jing, S.; Muqing, W.; Yong, B.; Min, Z. An improved GAC routing algorithm based on SDN. In Proceedings of the 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 13–16 September 2017; pp. 173–176. [CrossRef]

35. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* 2006, 1, 28–39. [CrossRef]

36. Shreya, T.; Mulla, M.M.; Shinde, S.; Narayan, D.G. Ant Colony Optimization-based Dynamic Routing in Software Defined Networks. In Proceedings of the 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 1–3 July 2020; pp. 1–7. [CrossRef]

37. Linfeng, Y. Research on SDN Load Balancing Based on Ant Colony Optimization Algorithm. Master’s Thesis, Harbin Engineering University, Harbin, China, 2019.

38. OpenFlow Switch Specification. ON Foundation. 2011. Available online: https://www.opennetworking.org/ (accessed on 1 February 2021).

39. Lantz, B.; Heller, B.; McKeown, N. A network in a laptop: Rapid proto-typing for software-defined networks. In Proceedings of the 9th ACM SIG-COMM Workshop on Hot Topics in Networks, Monterey, CA, USA, 20–21 October 2010; ACM Press: New York, NY, USA, 2010; p. 19.

40. Thakur, N.; Han, C.Y. An Ambient Intelligence-Based Human Behavior Monitoring Framework for Ubiquitous Environments. *Information* 2021, 12, 81. [CrossRef]