International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-1S5, December 2019

Single and Multi-Type Controllers with Soft Computing Methods and Routing in Software Defined Network

Mythrayee Ramasamy, Sanjay Pawar

Abstract: ***The software defined networking topology is enabled for efficient configuration for monitoring and enhance the network performance. The multi-controller placement becomes very critical, and it can be rectified by the proposed approach. The comparison of single type and the multi controllers are discussed in this paper.The first paper is compared with the dragonfly algorithm, in this we have considered only multi POX controllers. So the latency difference depends on the iteration of the algorithm. Thus the latency is reduced in proposed algorithm. Similarly the second paper has compared with the existing algorithm of firefly, which states the multi type of controllers such as POX, NOX, BEACON and FLOODLIGHT, among that latency is calculated and compared. The selection criteria** *are* **varied according to its type. Hence the routing is carried out on the two controllers alike POX, and NOX, through ad hoc on demand distance vector routing protocol. Subsequently, the selected controllers specifically are placed and simulated in Pareto-based Optimal Controller placement (POCO) tool for the Internet 2 topology and network simulator(NS2). Finally the comparison results makes the proposed system is better than previous methods.***

Keywords: ***Software defined network, multi controller placement,latency, firefly, dragonfly, ad hoc on demand distance vector routing protocol.***

I. INTRODUCTION

The dedicated devices to the software defined networking (SDN) can utilize the special kind of algorithms on order to manage the routing path, knowledge about the interconnection, and control the dataflow. Certain rules and the routing algorithms are regulated [1-3]. The four innovations behind the software defined networks are,

- Low power
- Flexibility
- Small size
- High performance [4].

The SDN controllers are used here to enhance the performance and manage the network in which the controllers are operated under server side, they can use the protocols to send the packets to the destination through the switches [5].

Manuscript published on 30 December 2019.
***** Correspondence Author (s)

Mythrayee Ramasamy, Department of Electronics and Communication, Usha Mittal Institute of Technology, (UMIT) SNDT University Mumbai, Maharashtra-400049.mythrayeeramasamy@gmail.com
Dr. Sanjay Pawar, Department of Electronics and Communication, Usha Mittal Institute of Technology, (UMIT) SNDT University Mumbai, Maharashtra-400049.

Along with software defined computer and software defined storage, IBM's software defined environment (SDE) allows for the automation and optimization of entire computing environments providing similar benefits [6-10]. The SDN controllers have the single and multi-type which can communicate with each other. When the failure occurs,

controllers can reassign the node to avoid the latency. The major problems in the software defined networks are

the controller placement, delay, power consumption, selection of controllers and load imbalance [11-15].

The layout of the proposed paper has been summarised as follows,in section I have the description of software defined networks, advantages, drawbacks, in section II have to gives the short term of recently developed technologies, in section III describes the problem and contribution of the work, in section IV describes the proposed methodology of the work, and finally the results are describes in section V.

II. LITERATURE REVIEW

The centralized controllers had been logically distributed over the devices used in the network. The reliability of the network was maintained by the determination of controllers count, and knows about the details of control action, for that the fault tolerant controllers had been developed by H.E. Egilmez*et al* [16]. So the control algorithms may arranged for the placement, and the algorithms were run under the 124 available network topologies.

The multi controller placement problem had been identified by F.J. Ros*et al* [17], in which the partition techniques were arranged for latency between the controllers and switches. There arises controller placement problem and the partition problem. In this case the K means algorithm was proposed to address the latency. Finally the latency is much more reduced (2.437times) from the average latency.

The reliability was focused on this work and it should be maximized while allowing the multi-controller placement. This can be characterized through expected percentage of control path loss, and it was developed by G. Wang, *et al* [18]. Thus the simulation analysis can ensure the reliability and the NP hardness. The multi-controller placement problem had been identified and it can be rectified through the POCO tool. This was the medium sized networks. Along this tool the heuristic approach was made for controller selection [19]. The SDN network decouples both the two planes alike data and control, these are going to reach the controller placement problem.

The heuristic algorithms had been developed by J. Liao*et al* [20]. Here were the newly approach of the density based controller placement had been developed, this can split the entire network into sub networks.

The size of the network had been selected through the capacity of the controller. Thus the approach can beat the latency, fault, and time consumption.

The key contributions in this research is summarized as follows,

- Design a software defined network (SDN) with a set of switches and links.
- An optimization algorithm for splitting the network and controller placement, considering the latency.
- A simulation based comparison of existing approach with the proposed approach to prove the performance with less number of controllers and propagation delay.
- The 34 nodes, and the 41 links are in the network topology, in which some nodes act as the controllers and remaining act as switches.
- The simulation is processed under the POCO tool and NS2.
- The routing is carried out in NS2 environment.
- Finally the performances are compared with the existing algorithms like Dragonfly, and Firefly.

III. PROPOSED METHODOLOGY

The above mentioned problems are rectified using the improved particle swarm optimization (IMPSO), and Multi-Objective ant lion optimization algorithm (MO-ALO). The clustering, and the controller selections are done under the algorithm. The performance of latency is compared with the existing algorithms like Dragonfly (DF), and firefly (FF), to shows the effectiveness of the proposed approaches. The SDN controllers are used to manage the flow control in order to enable the intelligent networking. It is based on some protocols to operate on the switches to send the packets. In first paper, the POX controllers are selected in the clustering. The communication is carried out between the switches and the controllers through the open flow controller. In second paper, the multi controller placement is the major problem, and it overcome through the usage of controllers like POX, BEACON, NOX and FLOODLIGHT. In both papers, the parameters of latency is evaluated. So here, we have to check the effectiveness of the proposed approach through the comparison performance.

Architecture of SDN

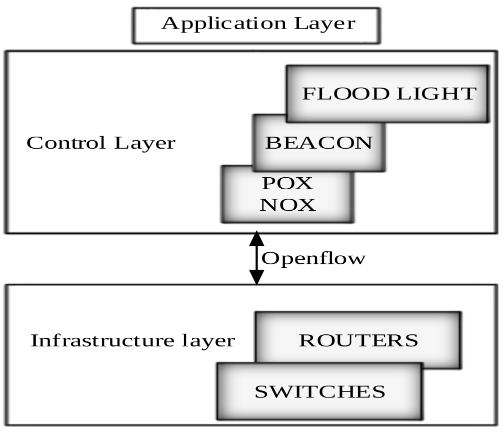


Fig 1: Architecture Of Software Defined Networks

This architecture can decouples the control action and forwarding functions, and it is shown in figure 1. The open flow protocol makes the solution to the SDN problem. The network performance and the monitoring have done through the network management and network configuration. The traditional networks are more flexible and troubleshooting. The control plane has several controllers, which is the brain of the SDN network where the whole intelligence is incorporated.

IV. NETWORK MODEL

The network model enables the switches, links, and controllers. Then the network have the clusters among that the controller selections are processed. The switches are represented as δ the links are ℓ, and the controllers are c, where $\delta = \lfloor \delta_1, \delta_2, \delta_3, \ldots, \delta_n \rfloor$, $\ell = \lfloor \ell_1, \ell_2, \ell_3, \ldots, \ell_n \rfloor$, and $\Phi = \lfloor \theta_1, \theta_2, \theta_3, \ldots, \theta_c \rfloor$. The clusters have to satisfy the following constraints like [22],

$$\varphi = (\theta_j) \neq 0 \tag{1}$$

Where $j = 1,2,3,\ldots,c$

$$\varphi(\theta_j) \bigcap \varphi(\theta_k) = 0 \tag{2}$$

Where $j \neq k, \quad j,k = 1,2,3,\ldots,c$

The cluster partition is done with the load capacity, controller power, switch and it is given by,

$$\sum_{\delta_k \in \varphi(\theta_j)} \rho(\delta_k) \leq LD(\theta_j), \forall \delta_k \in \varphi(\theta_j), \forall \theta_j \in \Phi \tag{3}$$

Generally, heavy load controllers have high failure probability as those controllers only have little resource to manage the errors. In such cases, propagation latency will increase, thus to rise in the resilience against this event, the optimization problem not only observe the latencies during failure-free routing, but also consider the worst case latencies during controller failures.While placing the controller on a particular location, it should not exceed constraint of the maximum, then the controller-to-switch propagation delay and it is calculated by,

$$Latency = \frac{Number\,of\ iteration}{Number\,of\ controllers} \tag{4}$$

Controller selection

Dragonfly Algorithm

The swarm intelligence of dragon fly algorithms are adopted by two characteristics which are in static, and the dynamic swarming. The swarming phases are grouped by exploitation and exploration. The sub swarms can fly over the corresponding region which is named as static, and for dynamic swarming the flies move from one place to another.

The calculation of the latency, and minimization is done by this algorithm, the steps involved in the algorithms are given below, *a)* **Separation:**

The collisions are avoided by separation process, while doing so each fly can move separately to take the food source, and the expression is given by,

$$s_u = -\sum_{u=1}^{x} \delta - \delta_u \tag{5}$$

The above expressions are useful for avoiding the collision among the nodes, while transferring the useful information.

b) *Alignment:*The mismatching of speed is denoted by the alignment and it is obtained by,

$$a_u = \frac{\sum_{u=1}^{x} \varepsilon_u}{x} \tag{6}$$

In this step, each node has to maintain some energy level, and initially the nodes contain the energy.

c) *Cohesion:*The ability of the flies can move towards the food source and it is calculated by,

$$c_u = \frac{\sum_{u=1}^{x} \delta_u}{x} - \delta \tag{7}$$

The above expression states that the objective functions which decide the latency can be optimized using the algorithm.

Step 1:Initialize the number of nodes, energy, maximum number of iteration, upper and lower bound which are specified to the algorithm.

Step 2:The objective function have to be calculated, here the latency, single and multi controller placement are the main issues..

Step 3:Obtain the step vector and the position vector, for which the network model is designed in above section. The network model contains switches and controllers. The nodes can transmit the information over the switches.

Step4:The objective functions are not obtained properly by doing this, we have to update the step vector and the position vector by following,

$$\Delta y_{t+1} = \left(Ss_u + Aa_u + Cc_u + F_u\left(ATS\right) + Ee_u\right) + G\Delta y_s \tag{8}$$

In equation (8) S is separation weight, A denotes alignment weight, and C is the cohesion weight; e_u is the position [enemy] of individual 'u'; E is the enemy factor and G is the inertia weight. The enemy factor e_u is calculated through,

$$e_u = \delta^- + \delta \tag{9}$$

Then the new position of each dragonflies are identified by using the below relation.

$$y_{t+1} = y_t + \Delta y_{t=1} \tag{10}$$

The above steps are repeated until reach the objective function

Pseudo code

```
**Start**
Initialize the switches, links, and the controllers
Initialize the load capacity
**Do** clustering
**While** satisfy the constraints are mentioned in equation (1)
**If**
{
The end condition is not satisfied
**Then** calculate the number of iteration and the controllers
}
Update the latency
**End**.
```

The single and multi-type controller placement problem is identified in this paper. So here considered the network topologies with some switches and links, in which the controller are selected to transmit the information via the neighbour nodes.The type of controllers just alike POX, BEACON, FLOOD LIGHT, and NOX. The dragonfly algorithm compared with the existing work while using the single type of controller, and the firefly algorithm can use the multi type controllers.

The structure of the software defined network is considered with number of nodes and edges which is represented as $g(n,e)$, the nodes are in terms of "n", and the bidirectional edges are presented to be "e". The number of the nodes in the network topology are represented as $k = |n|$. The number of controller are defined by,

$$P = \{P_1, P_2, ..., P_M\} \tag{11}$$

The switches are represented by,

$$Q = \{Q_1, Q_2, ..., Q_L\} \tag{12}$$

The matrix switches distribution matrix and is defined as $H = (h_{ij})_{M\times L}$.

$$h_{ij} = \begin{cases} 1, & Q_j \text{ is connected to } P_i \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Load capacity factors defined as the actual output to the average output and its controller is defined as δ. The load of the controller P_i can be estimated as eqn. (4),

$$\delta = \sum_{j\in Q_{P_i}} f_j(t) \times h_{ij} \tag{14}$$

Where $f_j(t)$ denotes the number of flow demands of switch Q_j at the time t. Therefore, the load set and the average controller load is evaluated by eqn. (5) and eqn. (6),

$$G_{total} = \sum_{i=1}^{M} \delta_i \tag{15}$$

$$\delta_{avg} = {G_{total}}/{M} \tag{16}$$

Therefore, the load capacity factor of the controller is derived from eqn. (7) as:

$$\delta = \frac{1}{M}\sum_{i=1}^{M}\left|\delta_i - \delta_{avg}\right| \quad (17)$$

In this the multi-controller placement is done through the dragonfly algorithm, while changing the controllers, the latency is being calculated. The process of routing is carried out, after the selection of multi-controllers with the help of dragonfly algorithm.

Firefly algorithm [24]

The fireflies are the insects which produce the rhythmic, and short flashes due to its process of bioluminescence. These lights can able to attract the partners of fireflies. All the fireflies move towards higher light intensity. The intensity of the light varied on the basis of distance, the light intensity decreases when the distance increases.

These algorithms exhibit three rules which are,

- Regardless of gender, the fireflies are attracted each other.
- The attractiveness of the fireflies correlates with the brightness of the fireflies
- The brightness of the fireflies depends on the objective function

The attractiveness and the distance is inversely proportional, and the relation is mentioned in below equation.

$$LI(r) = LI_0 e^{-\alpha r^2} \quad (18)$$

The attractiveness is characterized by,

$$\sigma = \sigma_0 e^{-\alpha r^2} \quad (19)$$

The distance between the links are calculated by,

$$r_{jk} = \left|y_j - y_k\right| \quad (20)$$

The movement of the firefly is given by,

$$\Delta y_j = \sigma_0 e^{-\alpha r_{jk}^2}\left(y_j - y_i\right) + \alpha e_i \quad (21)$$

Pseudo code

```
Start
Objective function: Multi controller placement
Generate the controllers POX, NOX, BEACON, and FLOOD LIGHT
Define the specified controllers
While (t<Maximum iteration)
For i=1: n all controllers
If ($(LI_j < LI_l)$)
Then
Move the controllers from j to l
Vary the attractiveness with the distance
End for
Rank the controllers and change the iterations
End while
Calculate the latency while changing the controllers
end
```

Routing

After the completion of the network model, the routing takes place. The transmission is done between any two controller like POX, NOX. The controller selection is based on the energy parameter. The packets are sent from the POX to NOX without any disturbance or collision. The transmission is being adopted with AODV protocol.

Ad hoc On Demand Distance Vector (AODV) Routing Protocol

The AODV protocol uses the three control strategies to maintain and obtain the routes which are,

- Request
- Reply
- Error

Route Request

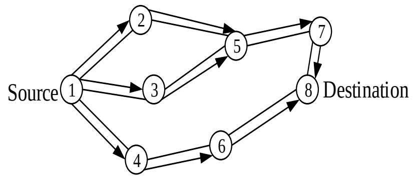


Figure 2: Route request

The above figure shows that the route request among the nodes, and the 1 is the source and 8 is the destination. The source send the request message to the destination.

Route Reply

The route reply is a method to deliver the destination information to the source node. The source can respond the message suddenly when after arrival of the data from the receiver, is shown in figure 3.

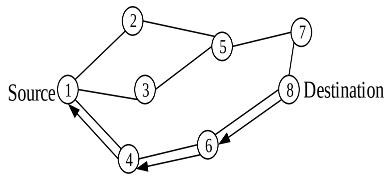


Figure 3: Route reply

Route Error

If the nodes are unable to send sufficient message to the sender, then the route error is generated by the receiver and it is forwarded to the sender.

The protocol is used to select the path to send, after founding the path, the packets are send from the source to destination, if error occurs, then it resend the packets to the source, so the error can be rectified.

V. RESULT AND DISCUSSION

The performance of the latency is compared with the existing algorithms like dragonfly algorithm and firefly algorithm. The latency is calculated with respect to the number of iterations and the number of controllers, when the presence of less controllers can give up the high latency and the presence of several controllers can reduce the latency. The simulation results are carried out under the MATLAB, POCO tool. The controllers such as POX, NOX, BEACON, and FLOOD LIGHT are used in the firefly algorithm and four POX controllers are used in the dragonfly algorithm.

The comparison results are taken out for the multi-controllers, even this can reduce the latency when compared with the existing, the clustering and the routing process are carried out in the NS2 environment.

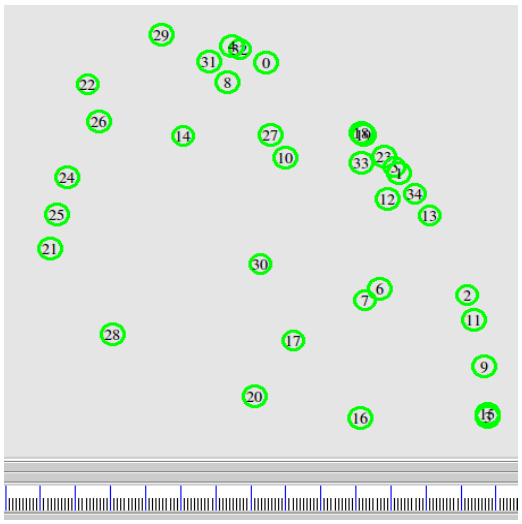


Figure 4: Randomized nodes

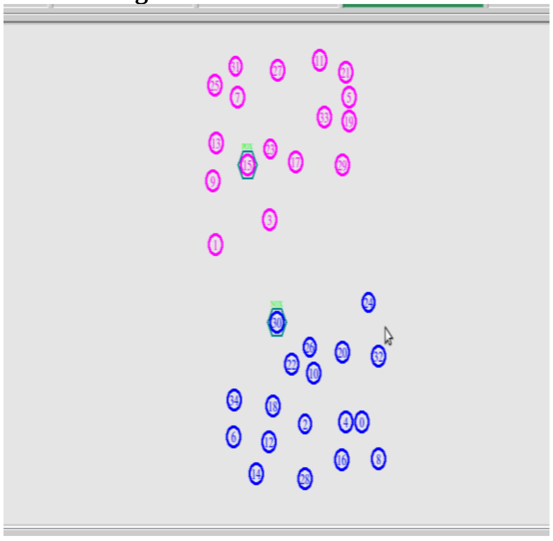


Figure 5: Clustering

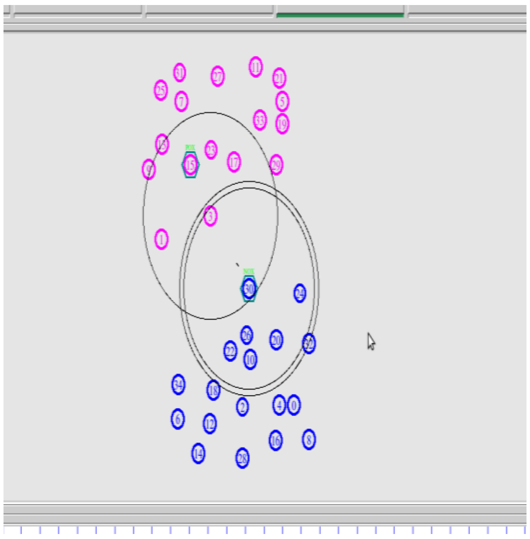


Figure 6: Routing

The above figure 4, 5, and 6 is a clustering process, initially randomly generating the kind of nodes and in which the clustering process is done on the basis of energy. The controller selection is the major process and in which the high energy is considered as a controllers like POX, and lower the energy is to be NOX. Hence the communication is carried out between two controllers POX and NOX through routing process. The routing is on the basis of ad-hoc on demand routing protocol. The number of packets are sent from source to receiver, with zero packet loss.

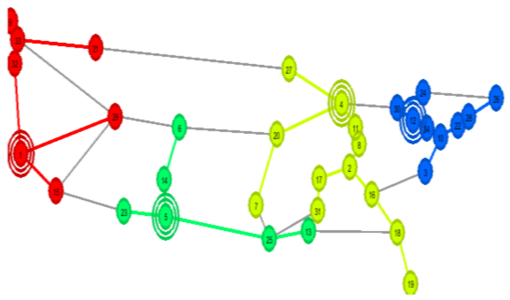
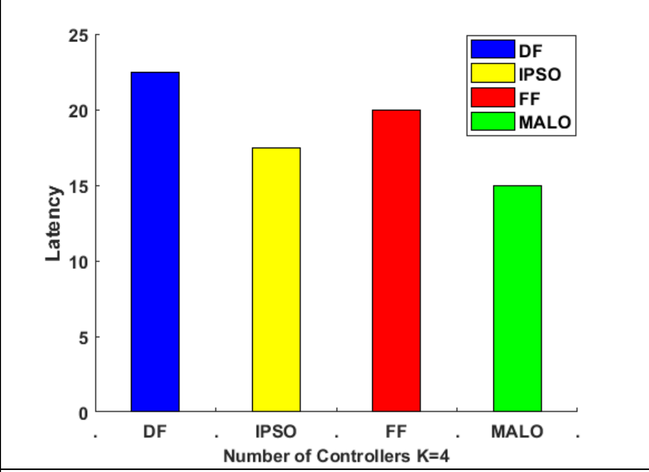
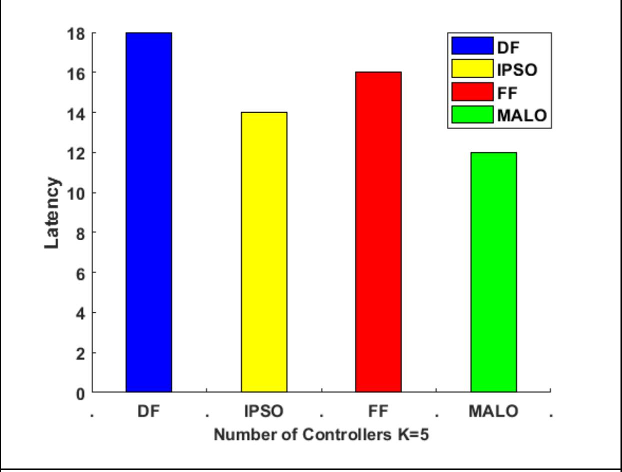
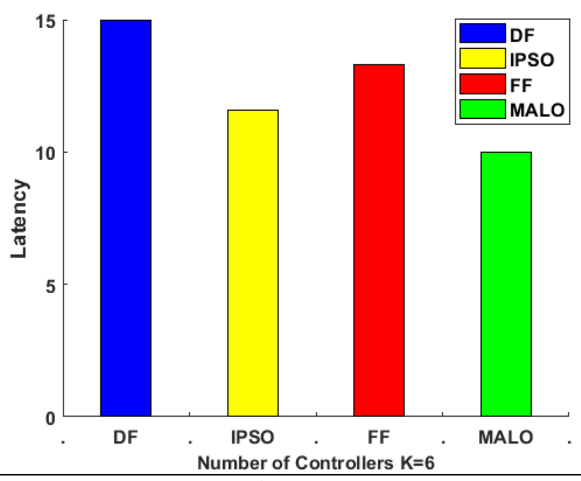


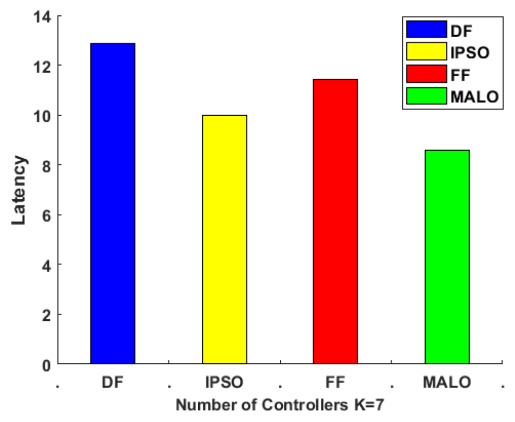
Figure 7: Controller selection by POCO tool

The above figure 7, shows that the controller selection by POCO tool, and the cluster heads are varied. This is for when k=4, and likewise the controllers are selected.

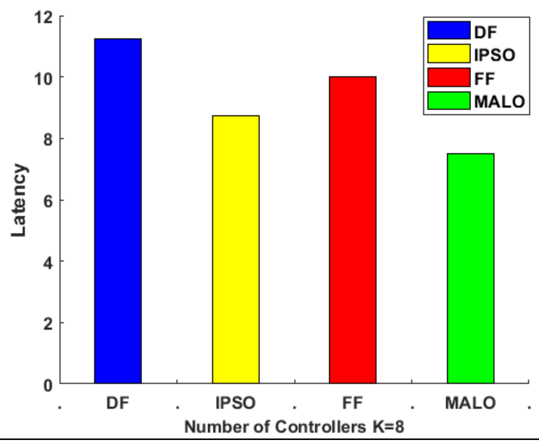
	
a) WHEN K=4	B)WHEN K=5



c) WHEN K=6



d) WHEN K=7



e) WHEN K=8

Figure 8: Comparison of multi controllersLatency with existing

The above figure 8, shows that the latency of corresponding controllers and it is compared with the dragonfly and firefly algorithm. By varying the controllers, the latency also varied. From the above comparison results easily obtain the low latency rather than the existing algorithm. Thus the proposed work shows the better performance.

Table 1: Comparison of latency with existing

	Existing		*Proposed*	
Latency	**DF**	**FF**	**IPSO**	**MALO**
K=4	22.5	20	17.5	15
K=5	18	16	14	12
K=6	15	13.3	11.6	10
K=7	12.85	11.42	10	8.57
K=8	11.25	10	8.75	7.5

The above table 1, shows that the latency of proposed and the existing works with the controllers like 4,5,6,7, and 8. By varying the number of controllers the latency is also changed. Thus the proposed system has the low latency when compared with the existing system.

Table 2:Algorithm notations

s_u	Separation
δ	Current position
δ_u	Neighbouring position
x	Number of neighbouring individual
ε_u	Neighbour velocity
q	Number of neighbour
LI	Light intensity
r	Distance between the links
α	Light absorption coefficient
LI_0	Original light intensity
σ	Attractiveness

VI. CONCLUSION

The Multi-controller placement problem is addressed and developed a new algorithm to reduce the latency and compared with the suitable control algorithms such as dragonfly, firefly.In this paper, the routing is carried out between POX, and NOX Controllers by adhoc on demand distance vector routing protocol. The controller selection criteria is based on the energy, which is given to the objective of the corresponding algorithm. The work addresses the NP hard problem, with the consideration of latency. The comparison is like IPSO-DF, and MALO-FF algorithm. The two objectives of single, and multi type controller is analysed by POCO tool, and the routing is carried out by network simulator (NS2). Thus the performance shows that the proposed algorithms have the effective way to lower the latency and effectively addresses the controller placement problem.

REFERENCES

1. R. Jainand S.Paul, “Network virtualization and software defined networking for cloud computing: a survey”, IEEE Communications Magazine, vol.51, no.11, pp.24-31, 2013.
2. M.Kobayashi, S.Seetharaman, G.Parulkar, G.Appenzeller, J.Little, J.V.Reijendam, P.Weissmann and N.McKeown,. “Maturing of OpenFlow and software-defined networking through deployments”, Computer Networks, vol.61, pp.151-175, 2014.
3. M.Banikazemi, D.Olshefski, A.Shaikh, J.Tracey and G.Wang, “Meridian: an SDN platform for cloud network services”, IEEE Communications Magazine, vol.51, no.2, pp.120-127, 2013.
4. H.Kim and N.Feamster, “Improving network management with software defined networking”, IEEE Communications Magazine, vol.51, no.2, pp.114-119, 2013.
5. M.Jammal, T.Singh, A.Shami, R.Asal and Y.Li, “Software defined networking: State of the art and research challenges”,Computer Networks, vol.72, pp.74-98, 2014.
6. C.Dixon, D. Olshefski, V. Jain, C.DeCusatis, W.Felter, J.Carter M.Banikazemi, V.Mann, J.M. Tracey and R.Recio, “Software defined networking to support the software defined environment”, IBM Journal of Research and Development, vol.58, no.2/3, pp.3-1, 2014.
7. C.E. Rothenberg, R.Chua, J.Bailey, M.Winter, C.N.Corrêa, S.C. de Lucena, M.R.Salvadorand T.D. Nadeau, “When open source meets network control planes”, Computer, vol.47, no.11, pp.46-54, 2014.
8. C.J. Bernardos, A.D.L.Oliva, P.Serrano, A.Banchs, L.M.Contreras, H.Jin and J.C.Zúñiga, “An architecture for software defined wireless networking”, IEEE wireless communications, vol.21, no.3, pp.52-61, 2014.
9. H.Ding, Y.F. Li, and S.K.Tso, “Dynamic optimization of redundant manipulators in worst case using recurrent neural networks”, Mechanism and Machine Theory, vol.35, no.1, pp.55-70, 2000.
10. M.Anan, A.Al-Fuqaha, N.Nasser, T.Y.Mu and H.Bustam, “Empowering networking research and experimentation through Software-Defined Networking”, Journal of Network and Computer Applications, vol.70, pp.140-155, 2016.
11. D.Kreutz, F.M.Ramos, P.E.Verissimo, C.E.Rothenberg, S.Azodolmolky and S.Uhlig, “Software-defined networking: A comprehensive survey”, Proceedings of the IEEE, vol.103, no.1, pp.14-76, 2015.
12. I. Monga, E.Pouyoul and C.Guok,. “Software-defined networking for Big-Data science-architectural models from campus to the WAN”, In High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion: pp. 1629-1635, November 2012.
13. S.Salsano, N.Blefari-Melazzi, A.Detti,G.Morabito and L.Veltri, “Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed”, Computer Networks, vol.57, no.16, pp.3207-3221, 2013.
14. S.H.Yeganeh, A.Tootoonchian and Y.Ganjali, “On scalability of software-defined networking”, IEEE Communications Magazine, vol.51, no.2, pp.136-141, 2013.
15. A.Hakiri, A. Gokhale, P Berthou, D.C. Schmidt and T. Gayraud, “Software-defined networking: Challenges and research opportunities for future internet”, Computer Networks, vol.75, pp.453-471. 2014.
16. H.E. Egilmez, S.T. Dane, K.T. Bagci and A.M.Tekalp, “OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks”, In Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), pp. 1-8,2012.
17. F.J. Ros, P.M. Ruiz,“On reliable controller placements in software-defined networks”, Computer Communications. Vol.77, pp. 41-51. 2016 Mar 1.
18. G.Wang, Y. Zhao, J. Huang, Q.Duan, J. Li. “A K-means-based network partition algorithm for controller placement in software defined network. InCommunications (ICC), 2016 IEEE International Conference on IEEE. pp. 1-6,2016 May 22
19. Y. Hu, W.Wang, X. Gong, X.Que, S.Cheng,“On reliability-optimized controller placement for software-defined networks”, China Communications. Vol.11, no.2, pp.38-54. 2014 Feb.
20. J. Liao, H. Sun, J. Wang, Q. Qi, K. Li, T.Li,“Density cluster based approach for controller placement problem in large-scale software defined networkings. Computer Networks, vol.112, pp.24-35, 2017 Jan 15.
21. H.Bo, W.Youke, W.Chuan'an, W.Ying,“The controller placement problem for software-defined networks”, InComputer and Communications (ICCC), 2016 2nd IEEE International Conference on IEEE. pp. 2435-2439,2016 Oct 14.
22. Liu, Shuai, H.Wang, S. Yi and F. Zhu, "NCPSO: a solution of the controller placement problem in software defined networks," In the proceedings of International Conference on Algorithms and Architectures for Parallel Processing, pp. 213-225, 2015.
23. S.Mirjalili. “Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective”, discrete, and multi-objective problems. Neural Computing and Applications. Vol.27, no.4, pp.1053-73, 2016 May 1
24. X.S. Yang,“Firefly algorithm, stochastic test functions and design optimisation”, International Journal of Bio-Inspired Computation. Vol.2, no.2, pp.78-84, 2010 Jan 1.