Open Access Article

OPTIMIZATION NETWORK ROUTING BY USING CUCKOO SEARCH

Salah Sabah Abed, Mohammad N. Fadhil
Department of Computer Science, University of Technology, Baghdad, Iraq
Cs.19.52@grad.uotechnology.edu.iq, Mohammad.n.fadhil@uotechnolohy.edu.iq

Abstract
Since the previous three decades, the field of computer networks has progressed significantly, from traditional static networks to dynamically designed architecture. The basic purpose of software defined networking (SDN) is to create a network that is open and programmable. Traditional network devices, such as routers and switches, may make routing decisions and forward packets; however, SDN divides these components into two planes, the control plane and the data plane, by splitting distinct components away. As a result, switches can only forward packets and cannot make routing decisions; routing decisions are made by the controller. OpenFlow is the communication interface between the switches and the controller. It's a protocol that allows the controller to identify the network packet's route across the switches. This project uses the SDN environment to implement the cuckoo search algorithm to determine the network's shortest path between two nodes. The cuckoo search algorithm was implemented using Ryu control.
Keywords: Mininet, Cuckoo Search, Multipath, Software defined networking
抽象的
过去三年以来，计算机网络领域取得了长足的进步，从传统的静态网络到动态设计的体系结构。软件定义网络 (SDN) **的基本目的是**创建一个开放且可编程的网络。传统的网络设备，例如路由器和交换机，可能会做出路由决策并转发数据包；然而，SDN **通**过将不同的组件分开，将这些组件分为两个平面，即控制平面和数据平面。结果，交换机只能转发数据包，不能做出路由决策；路由决策由控制器做出。 OpenFlow **是交**换机和控制器之间的通信接口。它是一种协议，允许控制器识别网络数据包在交换机上的路由。本项目使用SDN环**境**实现布谷鸟搜索算法，确定网络中两个节点之间的最短路径。布谷鸟搜索算法是使用 Ryu **控制**实现的
关键词：Mininet、Cuckoo Search、**多路径**、软件定义网络

1. Introduction

Optimal path selection necessitates continuous evaluation to topology of link quality to guarantee paths with superior performance. However, using end to end for measurements of ranking the dynamic path in large networks is efficient and scalable [1]. SDN have a global perspective of the topology and access to a significant network amount and variety, a data-driven approach is worth investigating. Some experiments have discovered correlations gleaned from network data, optimize resource, and

Received: December 29, 2021 / Revised: January 23, 2021 / Accepted: February 27, 2021 / Published: March 31, 2022
About the authors：Salah Sabah Abed
Email: Cs.19.52@grad.uotechnology.edu.iq

network performance [2, 3]. As a result, it's worth looking at a data-driven approach in which controller data is used to guide path found [4].

SDN opens up new possibilities for flexible routing packet management [5]. SDN divides the control plane of switches from the forwarding plane, allowing for remote and dynamic setting of forwarding tables. As a result, SDN accomplishes at least three major goals for traffic engineering across domains [6]: packet sending based on header information, remote forward rule configuration, and dynamic/programmatic forward rule configuration.

Meta-heuristic has many types one of them cuckoo search. The potential solutions are represented by the bird's eggs. The algorithm gradually replaces the bad solutions with new and better ones. The CS has a wide range of applications, including neural networks, job scheduling, and so on.

This study looks at the ability to find alternate dynamic path based on path attributes to see if it can help improve link utilization and network performance. This paper explains how Ubuntu may increase bandwidth usage and decrease latencies by employing SDN-based traffic engineering and using network measurements to accomplish dynamic path selection. The study also examines the utility of using network data received from an SDN and switch probing to apply the cuckoo algorithm to path found. Mininet is used to test an SDN based on network emulation, which uses cuckoo to disperse traffic across many forwarding links in order to maximize throughput and reduce latency.

The following are the work’s key contributions:

- A set of methods for computing a packet's expected path in a given setup, recording the actual path, and comparing the two paths to find a point of divergence.
- Aa prototype of the system's implementation.
- An experiment involving the insertion of both permanent and transient defects into network components.

The remainder of this article is structured in the following manner. In Section II, we go over the related study on routing optimization. In Section III, background theory for Mininet and cuckoo search algorithm and explain how to use it. Section IV outlines the model and formulates the problem, followed by Section V, which presents results and discusses performance evaluation. Finally, Section VI brings this article to a close and offers fresh areas for future research.

2. Related Work

Our strategy for achieving the objectives outlined in Section 2 is based on ideas offered in prior work in the fields of SDN network testing, verification, and debugging. Approaches to testing and verification aim to validate programs in terms of previously defined target invariants. Approaches to debugging are adapted to fix problems as they arise. We discuss related routing optimization research. Routing optimization techniques for typical distributed networks mostly focus on optimizing link weight. Fortz et al. [7] present optimization technique for intra-domain in routing based on IP addresses. To find the best link weight setting, they use a revised tabu search heuristic method. Similarly, Ericsson et al. [8] present a genetic algorithm (GA) to improve weight setting of link, while Srivastava et al. [9] suggest a Lagrange relaxation strategy to optimize classical network routing. Distributed networks in the traditional, on the other hand, force traffic to use the shortest channels and lack routing flexibility.

SDN for network operators may more easily monitor the flow routing in their networks and make timely changes to their route selection. Google [10] and Microsoft [11] have already solved linear programming in a lengthy series challenges to develop whole datacenter networks that can reach near %100 percent network utilization. These prior studies have focused on optimization routing in SDN. Agarwal et al. [12] suggest to deal with the issue of routing optimization. To balance traffic and improve the traffic splitting ratio routing, they offer Time Approximation (TA). Hu [13] and Wang et al. [14] create TA to improve traffic in SDN, similar to [12].

Guo et al. [15] provide heuristic strategies for optimizing the link path between nodes used SDN to enhance performance of network. Hong et al. [16] present a solution for gradual hybrid network deployment and routing optimization. Using the pick group table feature, they offer heuristic techniques to send flows on least path traffic. Jin et al. [17] present SDN for hybrid networks that allows for unified, fine-grained routing management. Chu et al. [18] present a method for quickly reacting to link failure in network. Routing optimization techniques in SDN allow route to send flows on path between Source and Destination.

To allow for fine-grained traffic control between sub-domains, Caria et al. [19] suggested to divide routing into numerous subdomains and deploy router in SDN. He et al. [20] present time approximation of polynomial approaches for TE issues in two hybrid modes, with an approximation of (1+). In a separate hybrid SDN situation, Xu et al. [21] optimize routing. In SDN can add switches to a standard network, they maximize incremental SDN rollout and flows routing together. Xu et al. [22] investigate entire SDN networks with traditional switching and SDN switching at all devices. Fibbing is a technique proposed by Vissicchio et al. [23] for centrally controlling link in routing by generating fake nodes to provide flexible routing. Hybrid network options are not the same as the hybrid SDN situation we looked into earlier.

3. Background Theory

3.1 Mininet emulator

Mininet is a network emulator software that precisely simulates the function and performance of any sort of forwarding element. SDN networks can be built to precise standards and tested on various network setups. We can migrate the SDN solution to a real physical network once the testing is completed on Mininet [24].

3.2 Cuckoo Search Algorithm

Cuckoo search algorithm (CS) is a cuckoo bird reproduction inspired algorithm [25]. It's crucial to correlate prospective solutions with cuckoo eggs while working with CS algorithms. Cuckoos usually lay their fertilized eggs in the nests of other cuckoos, hoping that their offspring will be nurtured by proxy parents. When cuckoos discover that the eggs in their nests do not belong to them, they either dump the foreign eggs out or quit the nests entirely. The three rules that govern the CS optimization method are as follows:

- Randomly choose a nest.
- Best nests will be passed down to the next generation with the finest egg quality.
- Construction a new CS depend on chance of $Pa \in [0,1]$.

The approximated can be replacing with a proportion *pa* of the *n* new nests by new solutions. The value of the objective function can easily be equivalent to the quality or fitness of a solution. The representation that is used in the implementation to represents a solution for each a nest. Carefully disregard the distinction between a nest. The goal is to replace a faulty solution with a fresh and perhaps superior solution.

Balance between global and local random walk is used, CS is effective for global optimization. A switching parameter pa ϵ [0,1] controls the balance between global and local optimization. We used Eq. (1) to define the global and local random walks, respectively. Table 1 lists their characteristics.

$$X_i^{t+1} = X_i^t + \alpha L(s, \lambda) \quad (1)$$

TABLE 1: PARAMETERS OF EQUATION 1

Parameter	Description
X_i^t	Random permutation determines the current position.
α	Scaling factor for position step size
S	step size
L	The Lévy used to determine the magnitude of a random walk's steps.

The Cuckoo search algorithm's core phases, which are based on 3 principles, can be stated as algorithm 1. Animals in the wild look for food in a random fashion. prediction move for both locations based on probability to predict next location this probability is random walk. The chosen direction is implicitly determined by a probability that may be statistically represented. Many animals and insects have been shown to exhibit the typical characteristics of Lévy flights, according to various research [26]. In this work used distribution probiplity which is levy function for random walk. The distance from the random walk's origin tends to a stable distribution after a large number of steps.

Algorithm 1: Cuckoo Search

1 Fitness-function $f(x) = (x_1, x_2, x_3, \ldots, x_d)^T$
2 Initializing *n* nests, *where* $x_i (i \leq n)$.
3 **While** (t<Max-generation-value) **do**
4 Levy flights use to update x_i
5 Get its fitness F_i
6 Pick a nest x_j at random
7 **If** ($F_i > F_j$) **then**
8 Replace x_j by the new solution
9 Rest worse nest use Eq. (1)
10 Rank the nest (solution) and keep the best nest
11 Final result output and presentation

The fitness function determines the best communication path. The value of the fitness function is affected by distance and delay. The best communication line is picked as the one with the fewest nodes and the shortest distance.

3. Proposed Model

The proposed multipath selection paradigm for routing in SDN is discussed in this section. Cuckoo search method has many advantages that is why suggested for path routing selection surpasses the present routing path. The path is found using the cuckoo search method. The benefits of the cuckoo search are considered while selecting the best path from many paths K revealed during the path discovery technique. When compared to other metaheuristic algorithms, the Cuckoo search algorithm has the following advantages: it is more versatile and robust for various optimization problems. It may readily be expanded to investigate optimization with several problems with a wide range of constraints. The suggested method introduces fitness function that takes into account several metrics such as distance and delay, resulting in improved performance with minimal latency. Figure 1 shows the proposed model for path selection in SDN, which is based on the cuckoo search.

Based on the cuckoo search, this article presents a path method for SDN networks. The following three steps make up the implementation process:

- The supplied dynamic environment is used to initialize the nodes.
- It is identified which pathways lead from the source to the target node.
- The fitness function and the cuckoo search algorithm are used to pick multipaths.

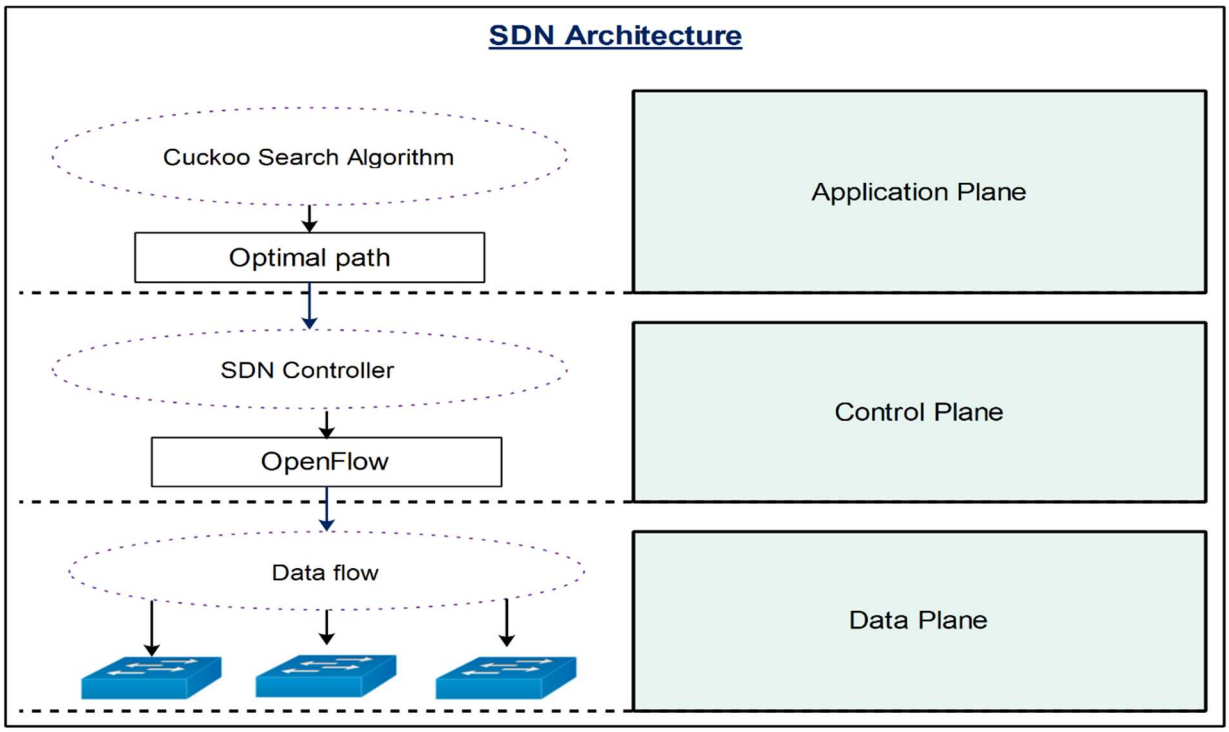


FIGURE 1: PROPOSED PATH SELECTION MODEL BASED ON CUCKOO SEARCH IN SDN

Figure 1 shows how to used cuckoo search by the application layer to retrieve the multipath topology and status of each path. The predicted multiple performance factors in the OpenFlow select group table

based on bucket value, ensuring that traffic on the network is spread across all feasible paths based on the path value. Simulation results show that this technique can successfully increase multipath resource usage in SDN networks, dramatically enhance traffic transmission, and achieve multipath. In SDN, the cuckoo search algorithm is represented in Figure 2.

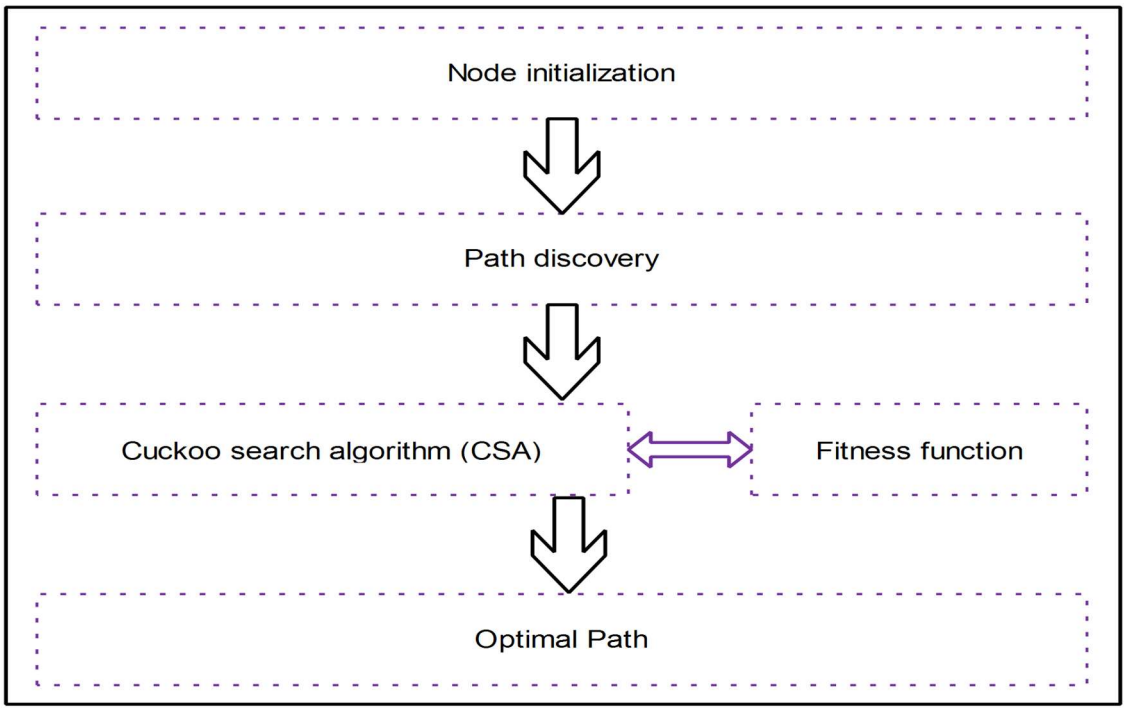


FIGURE 2: CSA BASED ON SDN

3.1 Node initialization

Because of its numerical control separation and programmability qualities, SDN allows to acquire and manage our topology. The Switches module in the SDN controller implements the topology discovery and management mechanism by a packet being sent out a packet carrying the underlying network's connection layer (OpenFlow). Following receipt of the link layer, the switch (OpenFlow) sends a packet to the controller carrying link information between switches, and the SDN finally detects and maintains our topology based on the link discovery protocol's feedback information. A connection discovery approach like this consumes a lot of communication traffic and incurs a delay penalty.

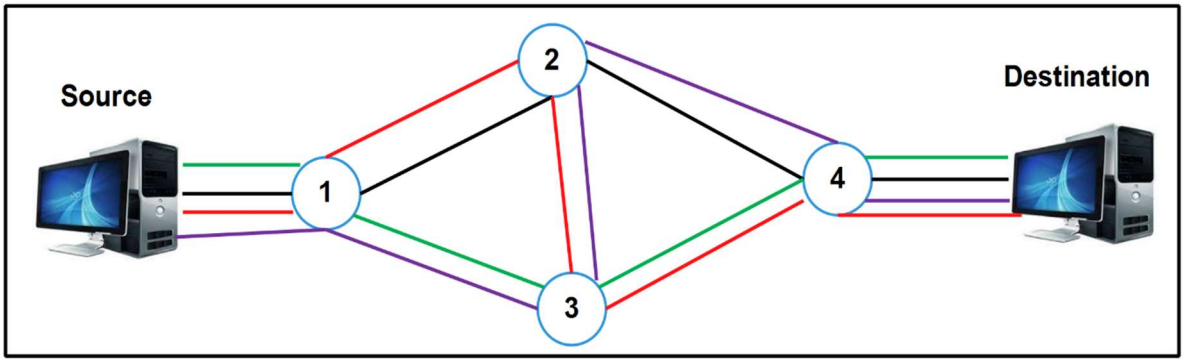
The SDN's nodes are set up in the dynamic environment that has been configured. The nodes serve as both a switch and a router. The network region is where the nodes are connected. The nodes' coordinates are computed, allowing the nodes' location and velocity to be determined in the future. Let's say there are m nodes in the network topology, which is given by $1 < i < m$.

3.2. Path discovery

This work employs the cuckoo search method to implement path topology discovery, which reduces telecommunication and reduces the value of delay. Consider the topology of our network is *G (h, s),* where *h* denotes the number of hosts and *s* denotes the number of switches (OpenFlow) where is setting by SDN. Let *P* be the total number of paths connecting the source and destination nodes.

$1 < j < P$ is the solution. The cuckoo search path search algorithm is shown in algorithm 1 to identify all possible multipath between any two hosts. SDN used to retrieve the whole topology and link connection states, and give usable information for further load balancing, by implementing this approach.

Figure 3 depicts the path found between the Source and Destination nodes in a network topology where *m* used 4 nodes. The nodes are 1, 2, 3, and 4. The communication between the point of source and the point of destination is accomplished through several pathways. The path for data packet transmission is initially discovered based on the connectivity present between the nodes. The link between Nodes 2 and 3 increases the number of paths revealed in addition to the direct connectivity between surrounding nodes. Four pathways are discovered between the source and the destination. 1-2-4, 1-2-3-4, 1-3-2-4, and 1-3-4 are the numbers. Utilizing the suggested cuckoo search method, the multipaths suitable for communication between the hosts are picked from the path found using the existing link between the nodes.



Color	Path	Cost
———	1-2-4	3
———	1-2-3-4	4
———	1-3-2-4	4
———	1-3-4	3

FIGURE 3: PATH DISCOVERY

Multiple pathways are present between the Source and Destination nodes in network topology, depending on the link associated with the channel. The connection between the Source and the Distance takes place across a number of paths. *K* is the number of pathways detected for links (*L*) existent in nodes linked with *m* nodes between the Source and Distance.

3.3. Implement cuckoo search algorithm

Multipath selection utilizing the suggested cuckoo search and the fitness function is discussed in this section. The cuckoo search algorithm is broken down into steps below.

Step 1: In this step initialized nest (host) at randomly where *K* is path detected cost of nest (host). Let assume *n* initial population for nest's (host) as shows in Eq. (2). Our goal to find and select best path among number of *K* paths.

$$f_y = f_1, f_2, f_3, \dots, f_g \quad (2)$$

The population count is assumed to be c in addition to the host. The count value is incremented by one for each iteration.

Step 2: Using levy flight, generate a new solution (host nest) at random according to equation (1).

Step 3: Pick a nest (host) at random from Eq (2). The chosen random solution called f_y.

Step 4: In this step used fitness function for make evaluation and randomly chose nest (solution).

$$f_d = \min(p_d) \quad (3)$$

Assume f_d generated solution and f_y random solution from the nest as shown in equations (4) and (5).

$$fitness(f_d) = f_{d(fit)} \quad (4)$$

$$fitness(f_y) = f_{y(fit)} \quad (5)$$

where f_y chosen solution from random nest and f_d generated solution.

Step 5: Evaluate each nest by used fitness and discarded the worst nest. Equation (6) determines the optimal answer.

$$fit(f_d) < fit(f_y) \quad (6)$$

Step 6: The process of searching continues until the number *c* reaches its highest point. For SDN routing, the solution with the greatest number is chosen as the best output solution.

4. Experiment and Analysis

This part conducts simulation experiments on this method in order to verify the previously presented model. The operating system in this experiment is Ubuntu 16.04, the network simulation software is Mininet, and the SDN controller is Ryu software. Ryu and Mininet are installed on a PC 2.60GHz Intel i7 9750 processor, 16GB RAM, and a 64-bit Ubuntu operating system. The Ryu controller runs on 64-bit Python 3.7 as its operating system. The proposed paradigm in this paper is implemented using the Python programming language to create Ryu controller application files.

Different topologies are employed in the experimental evaluation to demonstrate the performance of our suggested model. We'll start with the environment setting in this part. Extensive tests are then used to establish SDN node deployment percentage and path selection. After that, we show how well the suggested model performs in terms of cost minimization using a predetermined SDN node deployment percentage and path selection. Suddenly, we show how long the suggested model takes to compute. The parameters for cuckoo search are listed in Table 1.

TABLE 2: CUCKOO SEARCH PARAMETERS

Parameters	Value
Iteration	30
Nest	15
Alpha	0.1
Beta	1.5
Param	0.25

We must first establish the number and location in order of SDN nodes to acquire SDN. Mininet is used to build a network topology with two hosts, six switches, and a controller, as shown in Figure 4. Traffic is routed with path selection limitations using the Cuckoo search algorithm.

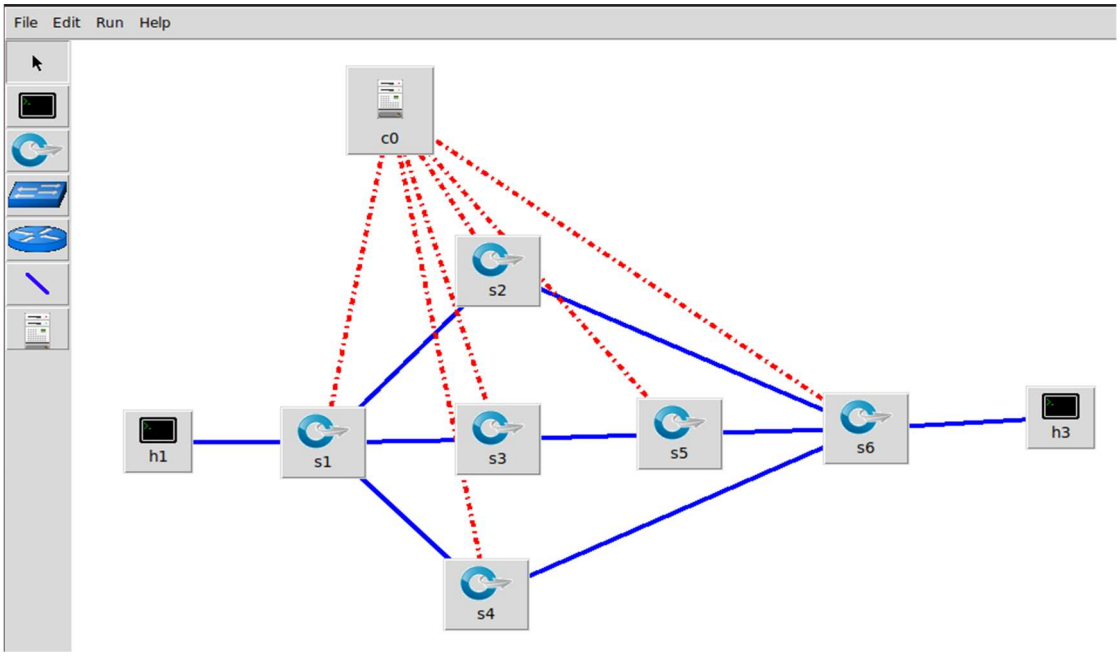


FIGURE 4: NETWORK TOPOLOGY 1

Figure 5 shows SDN application "app.py" is started the launch function is what it called, which returns information about the Ryu controller and whether it is operating or not. It also listens for requests the topology of network is built using a python script on port 6633 (which can be altered).

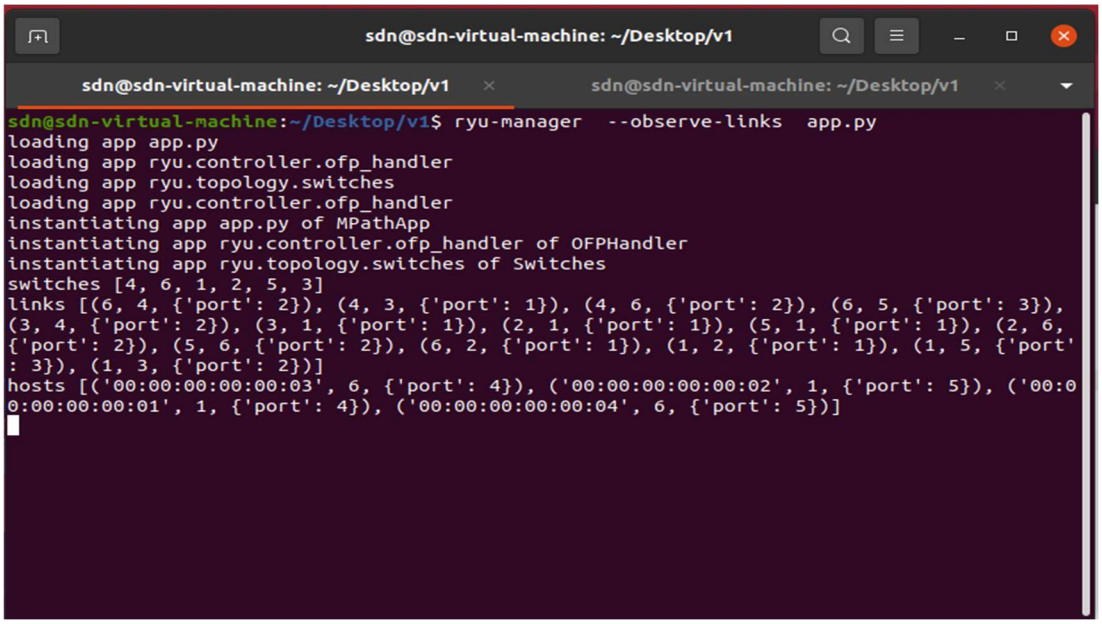


FIGURE 5: RUNNING RYU APP

A "Switch connection event" is generated whenever a switch is added to the network after running a python script called "topo.py" containing the network's requirements, as seen in Figure 6.

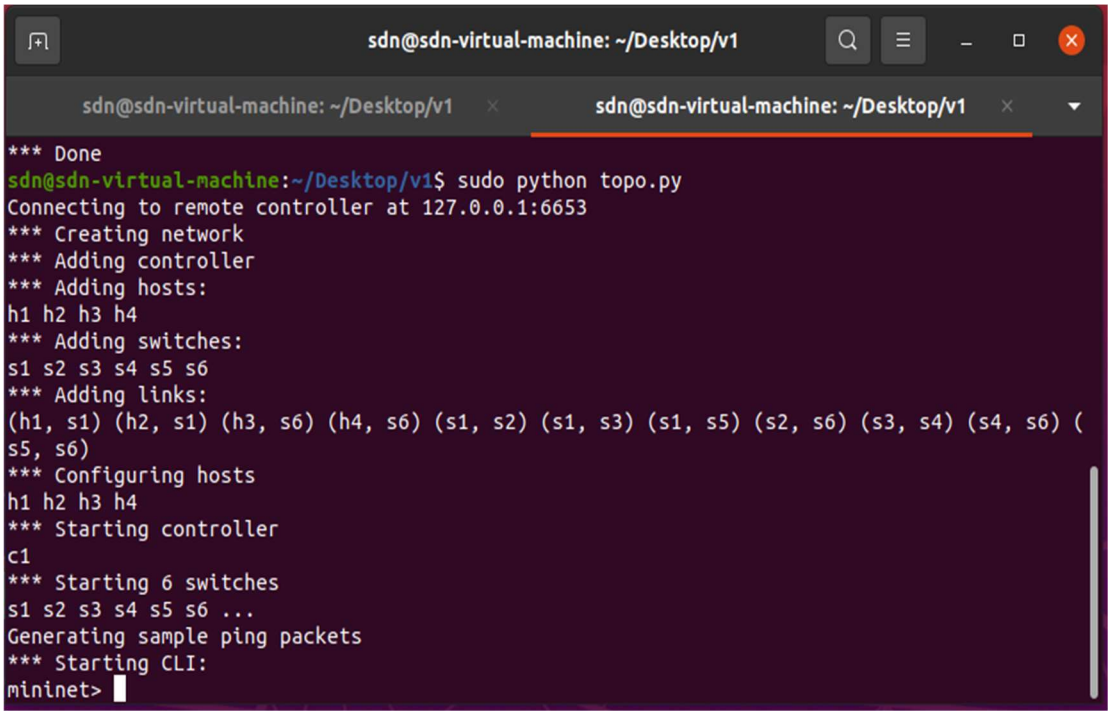


FIGURE 6: SWITCHES AND LINKS ARE BEING ADDED

Packet transmission can commence after all of the switches and links have been identified. The cuckoo search algorithm discovers the shortest channel for transmission when host "h1" pings "h3," as seen in Figure 7. The packets are sent along the path one by one. The IP addresses of all network devices involved, as well as the flow tables of the switches, are discovered via an ARP packet.

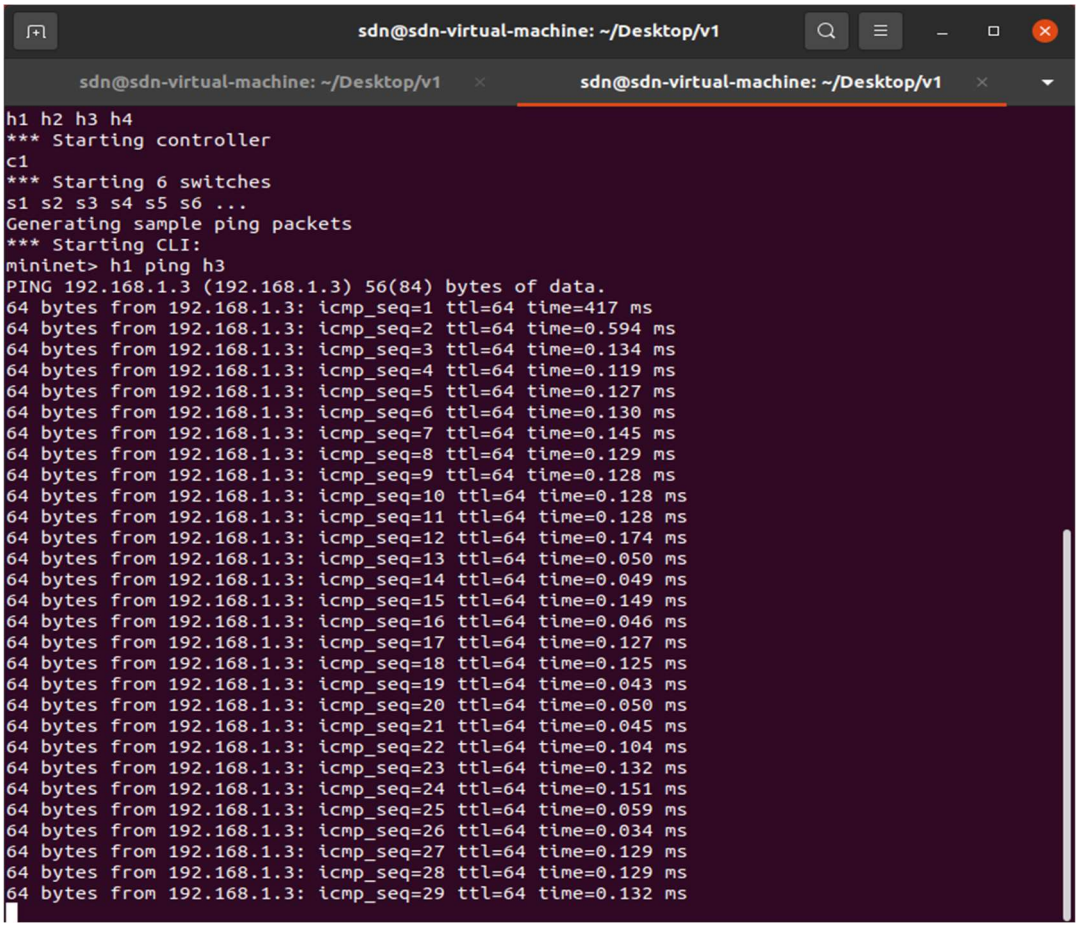


FIGURE 7: PING FROM HOST 1 TO HOST 3

Table 3 depicts the path that was chosen from topology one when it was formed. This scenario involves sending a message from Host 1 to Host 3 and determining the optimum path.

TABLE 3: PATH DISCOVERY AND SELECTION

Path discovery	Hops	Path selection
S1→S2→S6	3	√
S1→S3→S5→S6	4	×
S1→S4→S6	3	√

Figure 8 depicts two hosts, h1 and h2, as well as six switches (S_1, S_2, S3, S_4, S_5, and S_6) and a controller. In the diagram below, a new link is added between s2 and s3, as well as between s3 and s4.

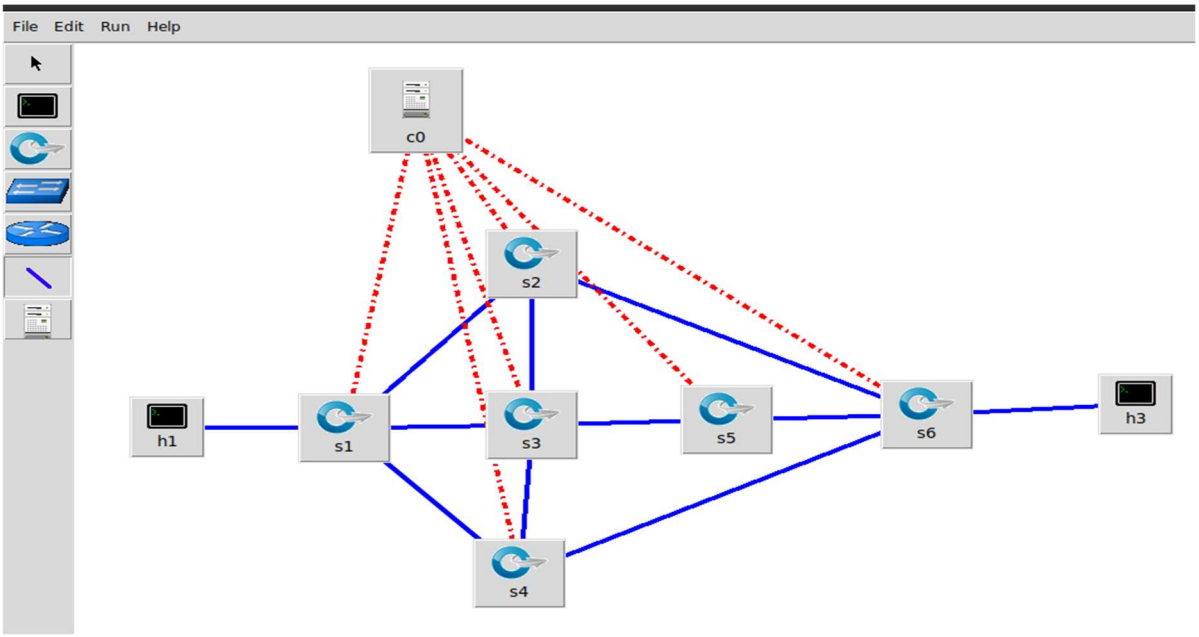


FIGURE 8: NETWORK TOPOLOGY 2

Table 4 depicts the path that was chosen from topology one when it was formed. This scenario involves sending a message from Host 1 to Host 3 and determining the optimum path.

TABLE 4: PATH DISCOVERY AND SELECTION

Path discovery	Hops	Path selection
S1→S2→S6	3	√
S1→S3→S5→S6	4	×
S1→S4→S6	3	√
S1→S2→S3→S5→S6	5	×
S1→S3→S2→S6	4	×
S1→S4→S3→S5→S6	5	×
S1→S4→S3→S2→S6	5	×
S1→S2→S3→S4→S6	5	×

Figure 9 depicts the cuckoo search algorithm's selection path. Figure 8 was utilized in the scenario to show the optimum path choice.

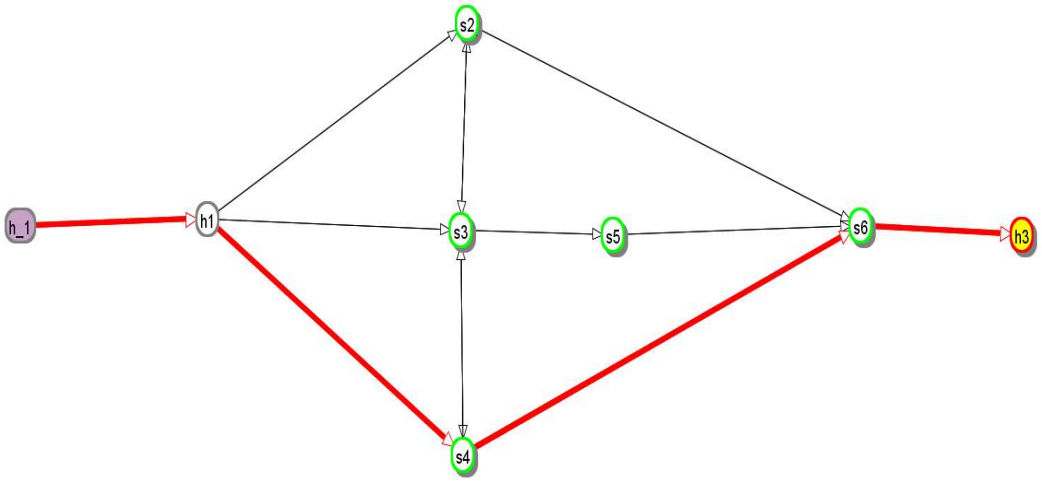


FIGURE 9: PATH SELECTION

Conclusion

To illustrate dynamic programmability via controller, the cuckoo search algorithm was implemented in SDN environment using Ryu and Mininet in this study. The field of software defined networking is relatively new, yet it is rapidly expanding. Important research concerns, such as security and load balance, must be addressed. If a company requires a different form of network behavior, it might create or deploy a program to suit its requirements. This application could be a typical networking function like traffic engineering, policy routing, firewalls, or security. SDN has the ability to improve the efficiency with which network applications and services are deployed and managed. Load balancing and firewalling can be imitated in the future dependent on our campus needs. Because SDN allows developers to design applications based on individual campus requirements, such as during an online examination at a university, priority and greater bandwidth links can be provided to html sites during that hour. Load balancing can also be done using the link's cost and the number of controllers.

Reference

[1] A. Jain and J. Pasquale, "Internet distance prediction using node-pair geography," in *2012 IEEE 11th International Symposium on Network Computing and Applications*, 2012, pp. 71-78: IEEE.

[2] G. N. Rouskas *et al.*, "Choicenet: Network innovation through choice," in *2013 17th International Conference on Optical Networking Design and Modeling (ONDM)*, 2013, pp. 1-6: IEEE.

[3] T. H. Obaida and D. H. Abd, "A Robust Approach for Mixed Technique of Data Encryption Between DES and RC4 Algorithm," *Journal of Kufa for Mathematics and Computer Vol,* vol. 3, no. 2, pp. 48-54, 2016.

[4] H. Yin, Y. Jiang, C. Lin, Y. Luo, and Y. Liu, "Big data: transforming the design philosophy of future internet," *IEEE network,* vol. 28, no. 4, pp. 14-19, 2014.

[5] C. E. Rothenberg, M. R. Nascimento, M. R. Salvador, C. N. A. Corrêa, S. Cunha de Lucena, and R. Raszuk, "Revisiting routing control platforms with the eyes and muscles of software-

defined networking," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 13-18.

[6] A. Gupta *et al.*, "Sdx: A software defined internet exchange," *ACM SIGCOMM Computer Communication Review,* vol. 44, no. 4, pp. 551-562, 2014.

[7] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proceedings IEEE INFOCOM 2000. conference on computer communications. Nineteenth annual joint conference of the IEEE computer and communications societies (Cat. No. 00CH37064)*, 2000, vol. 2, pp. 519-528: IEEE.

[8] M. Ericsson, M. G. C. Resende, and P. M. Pardalos, "A genetic algorithm for the weight setting problem in OSPF routing," *Journal of combinatorial optimization,* vol. 6, no. 3, pp. 299-333, 2002.

[9] S. Srivastava, G. Agrawal, M. Pioro, and D. Medhi, "Determining link weight system under various objectives for OSPF networks using a Lagrangian relaxation-based approach," *IEEE transactions on Network and service management,* vol. 2, no. 1, pp. 9-18, 2005.

[10] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Computer Communication Review,* vol. 43, no. 4, pp. 3-14, 2013.

[11] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, 2013, pp. 15-26.

[12] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *2013 Proceedings IEEE INFOCOM*, 2013, pp. 2211-2219: IEEE.

[13] Y. Hu, W. Wang, X. Gong, X. Que, Y. Ma, and S. Cheng, "Maximizing network utilization in hybrid software-defined networks," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1-6: IEEE.

[14] W. Wang, W. He, and J. Su, "Boosting the benefits of hybrid SDN," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 2165-2170: IEEE.

[15] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in SDN/OSPF hybrid network," in *2014 IEEE 22nd International Conference on Network Protocols*, 2014, pp. 563-568: IEEE.

[16] D. K. Hong, Y. Ma, S. Banerjee, and Z. M. Mao, "Incremental deployment of SDN in hybrid enterprise and ISP networks," in *Proceedings of the Symposium on SDN Research*, 2016, pp. 1-7.

[17] C. Jin, C. Lumezanu, Q. Xu, H. Mekky, Z.-L. Zhang, and G. Jiang, "Magneto: Unified fine-grained path control in legacy and openflow hybrid networks," in *Proceedings of the Symposium on SDN Research*, 2017, pp. 75-87.

[18] C.-Y. Chu, K. Xi, M. Luo, and H. J. Chao, "Congestion-aware single link failure recovery in hybrid SDN networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 1086-1094: IEEE.

[19] M. Caria, T. Das, A. Jukan, and M. Hoffmann, "Divide and conquer: Partitioning OSPF networks with SDN," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 467-474: IEEE.

[20] J. He and W. Song, "Achieving near-optimal traffic engineering in hybrid software defined networks," in *2015 IFIP Networking Conference (IFIP Networking)*, 2015, pp. 1-9: IEEE.

[21] H. Xu, J. Fan, J. Wu, C. Qiao, and L. Huang, "Joint deployment and routing in hybrid SDNs," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, 2017, pp. 1-10: IEEE.

[22] H. Xu, H. Huang, S. Chen, and G. Zhao, "Scalable software-defined networking through hybrid switching," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, 2017, pp. 1-9: IEEE.

[23] S. Vissicchio, L. Vanbever, and J. Rexford, "Sweet little lies: Fake topologies for flexible routing," in *proceedings of the 13th ACM Workshop on Hot Topics in Networks*, 2014, pp. 1-7.

[24] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp. 1-6.

[25] X.-S. Yang, *Nature-inspired optimization algorithms*. Academic Press, 2020.

[26] C. T. Brown, L. S. Liebovitch, and R. Glendon, "Lévy flights in Dobe Ju/'hoansi foraging patterns," *Human Ecology,* vol. 35, no. 1, pp. 129-138, 2007.