Journal of Namibian Studies, 33 S3(2023): 2536-2553 ISSN: 2197-5523 (online)

Optimization Flow Control In Software-Defined Networking Using Cuckoo Search Algorithm

Salah S. Abed [1,] *, Mohammad N. Fadhil [2]

[1] Department of Computer Science ,University of Technology , Baghdad, Iraq.

[2] Department of Computer Science, University of Technology, Baghdad, Iraq.

* cs.19.52@grad.uotechnology.edu.iq

Abstract:

Network managers can scale their networks more easily and experience fewer issues by using software-defined networking. They have evolving requirements over time, just like big scale networks like data centers. The speed at which data is transferred between nodes and the expense of the physical equipment are both very important. This thesis offers a framework for avoiding network momentum by using the cuckoo search algorithm to find the best path at the lowest cost, as well as for lowering the cost of the network's physical equipment when it needs to be expanded and for lowering the time it takes to send packets after finding the shortest path as the network is built in different sizes. It is generated automatically using the Spanning Tree Protocol. (STP), where the virtual ring-free network is built using data connection layer of the OSI model system, and the paths are then computed using the paths calculation algorithm and the cost of each route. The fitness function, one of the steps in utilizing the cuckoo search method is used to ascertain the best path, which primarily relies on time and distance factors.

Keywords: Mininet, Cuckoo Search, Multipath, Software defined networking.

1. Introduction

Both the use of and the extent of the internet kept growing. Networks are getting bigger, and datacenters are getting bigger and more powerful. In order to keep up with the growing volume of data flows, this growth is necessary. The annual amount of internet protocol traffic will exceed one zettabyte in 2016 and two zettabytes in 2019[1]. The networks must load-balance and multipath in order to handle the additional traffic and maintain high speeds. Due to their difficulty in installation and upkeep, traditional Ethernet switches have a number of disadvantages [2]. On the control plane of conventional network devices, network device

manufacturers usually implement standardization protocols developed by groups like the Internet Engineering Task Force (IETF) and Institute of

Electrical and Electronics Engineers (IEEE). The network can now be partially controlled by the
Simple Network Management Protocol (SNMP), Plymouth-Canton Educational Park (PCEP), and Network Configuration Protocol (Netconf) [3]. The software define network (SDN) paradigm offers a way to build and manage networks more effectively [4], [5]. A brand-new networking technology called software define networks seeks to address long-standing networking problems.

It is then moved to a dynamic, programmable software-defined network controller, or control plane. An excellent example of SDN is OpenFlow, which will be covered in more detail in the part after this one[6]. They prioritized traffic analysis over dynamic load-balancing and multilayered networks, despite their prior interest in the areas of load balancing and multi-paths [7].

The purpose of this thesis is to use SDN approaches to create a network architecture and choose the best path by investigating load balancing and multipaths to aid in network performance decisions. The suggested solution specifically addresses SDN and the use of multipath to create a novel decision-making process. This thesis also introduces a novel framework for multipath options based on the cuckoo algorithm, with the purpose of improving fault tolerance, security, and network monitoring. The following are the thesis's primary objectives:

- To build several topologies in order to test our technique.
- The cuckoo search algorithm is used to identify the best path between the source and the destination.
- To enhance load balance and make path selection more dynamic.
- SDN will be used for multipath and load balancing.

2. LITERATURE REVIEW

This part provides a review of the literature on load-balancing traffic flows through multipaths . The evaluation of the papers is based on their applicability to the present research. The limits (reaction time or/and throughput) in the related literature are also listed in a summary at the conclusion of this section. Wu et al.'s adaptive multipath selection based on SDN was suggested in 2018[8]. Three factors for this algorithm were proposed: bandwidth, packet loss, and time delay. All of these settings are built on SDN for centralized control. The algorithm utilized OpenFlow control and produced excellent accuracy. The authors determined the ideal path and used it in the suggested algorithm. A crucial application was

having trouble because of the latency issue with the multipath routing method, according to Aldwyan and Sinnott 2019[9]. They came up with fresh ideas for how to make applications more accountable for providing latency while being conscious of a broken link. For the optimal path selection in the datacenter, they recommended an SDN-based algorithm. In their operations, they made use of container innovations and highly scalable application design. The method reduces switchover time by making available distribution methods for microservices their overlapping positioning across different datacenter networks, enabling latency-aware resilience choices autonomously. The result of the application was a reduction in connection 40 milliseconds for processing time.

2019, Hsieh and Wang[10], The timeout problem SDN multipath for routing was found when the primary connection failed and the packet data did not arrive at its destination in a timely manner. The writers suggested a workable technique for locating and resolving connection errors. They provided a straightforward method for locating and fixing communication errors. The suggested remedy is built on a multicentral SDN architecture. Using cost of the connection weighting using load standard division and switch controller transmission delay, the topology's local

and global controllers determine the best path. The trial's results show that the employed technique can handle a bad connection with a delay of just 7 milliseconds.

Diego et al. 2020[11] suggested a brand-new approach for the protocol's route discovery. In the suggested approach, multipath based on a single network with path selection was used to find multiple paths that were present on the network. The suggested approaches, known as One-Shot Multiple Disjoint Path Discovery Protocol (1S-MDP), demonstrate faster execution times and a reduction in the number of disjoint pathways found. 2020[12], Hemin Kamal and Miran Taha, proposed new SDN-based multi-path routing solution . This technique sent data over multiple paths with a time reduction. This method chooses the most direct route to the originator to the reserve based on the size of the packet and the connectivity of the network . This approach is effective at speeding up the handling of the connection. As a result, they eliminated the time-consuming aspect of path selection in the method they suggested.

They omitted a multipath routing algorithm from their papers, though. Our approach varies from theirs in that it takes into account variables for the conduit that connects the source and the target be optimized.

3. BACKGROUND

A. Traditional Network

The two fundamental parts of a normal network are the control plane and the data plane. The control plane defines the connections that will be made, the data forwarding, the rules that will be used, and the route that will be followed[13]. Because it utilizes an end-to-end connection between the host and the server to transfer data from one channel to another, the abstract layer may refer to it as a data plane[14]. As data traffic and network modifications have increased, it has become more important than ever for the network to respond promptly and effectively[15]. The idea of how machines exchange information through services without involving humans has undergone a radical transformation. The same ideas are now being successfully applied to network resource virtualization[16].

B. Software Defined Network (SDN)

Network administration has been made incredibly simple by the new networking paradigm, known as the Software Defined Network approach. By providing a customizable, adaptable interface that manages the operations of the entire network, it also promotes innovation in networks. On the other hand, traditional networks are thought to be difficult to use, prone to errors, and slow to add new features. In this paper, we discuss the SDN's theoretical underpinnings and applications[17]. The sole node used by SDN to consolidate control is the "Network Controller". Devices no longer control networks; they now only forward traffic. The forwarding devices are programmed by the network administrator using OpenFlow[18].

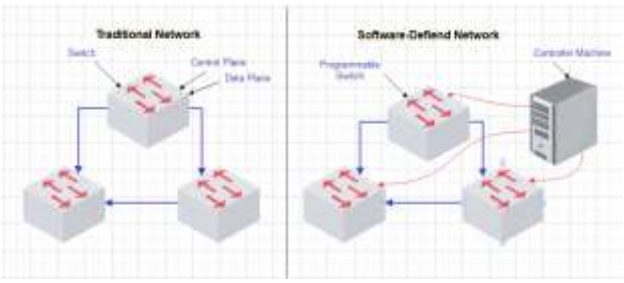


Figure1. Difference between SDN and the traditional network

C. Mininet

A simulation system called Mininet was created to support, evaluate, and imitate Software Define Network[19]. Both local and vast networks can support Mininet[20], [21]. Its main benefit is in facilitating collaborative network study by making self-contained SDN prototypes available for download and use by anyone with a PC or laptop[22], [23]. Without the need for a costly testbed, Mininet enables the creation of SDN applications and networks [24]. The Fire Safe Software Define Networking (Fs-SDN)

simulator, which employs a Raspberry Pi to create a cost-effective OpenFlow protocol tested for small SDN networking[25], is another important simulator. Mininet continues to be the preferred emulator for SDN networks even though some rival simulators have some simulation advantages over Mininet[26], [27]

4. PROPOSED SYSTEM

Many academics are interested in the subjects of dynamic load balancing and path selection. Finding suitable solutions for issues like route selection and dynamic load balancing is one part of the work on distribution networks. There is a lot of manual labor involved in choosing the best option and dynamically balancing the load. An example of a network topology issue relevant to this study is shown in Fig. 2. The restrictions might change based on the objectives and network design.

Figure2. Example of Network Topology

The above example shows sample network topology, in which the host (h1) sends a message to host (h3) and send data to the target via the specified path. This thesis aims to select the best path to send data and dynamic load balance means the sent data may be on path (h1→S1→S3→S5→S6→h3) and the received data may be on another path, such as (h3→S6→S4→S1→h1). Given a set of switches S and a set of hosts H, the task is to identify the path selection using Eq. (1) for the topology.

$$P = f(H, S) \qquad (1)$$

Where P is route choice, and f is the process used to determine the most direct route between S when host H the communication is sent. Let us assume $S = \{S_i \mid S_i \text{ is switche } i \in \mathbb{Z}^+\}$ and $H = \{H_j \mid H_j \text{ is host}\}$. L creates a series of connections, which are joined by ports in S and H. The connection to harbor of $S_{i,n}$ to port of $S_{i,m}$ is denoted as $[S_{i,n}, S_{i,m}]$. When $n = m$, the connection is a switch's internal communication . S, if not, it serves as the exterior connection between S_n and S_m. Note that the connection is one-way . The switch connected to the link's first

segment is the source, and the switch connected to its second segment is the target. connecting the host and $H_{j,n}$ and host $H_{j,m}$ in topology S_i can be denoted as $[H_{j,n}: \text{port } x, H_{j,m}: \text{port } y]$, where x and y are, respectively, the amount of ports in hosts and switches.

5. RESULTS

A. Cuckoo Search Parameter

The cuckoo search criteria are established in this part and are shown from Table 1.

Parameters	Value2	6 Switches	12-Switches	15-Switches	20-Switches	30-Switches
Iteration	30	30	30	30	30	30
Nest	15	15	15	15	15	15
Alpha	0.1	0.1	0.1	0.1	0.1	0.1
Beta	1.5	1.5	1.5	1.5	1.5	1.5
Param	0.25	0.25	0.25	0.25	0.25	0.25

Table 1: Cuckoo search parameter

B. Creating Topology

In this study, five different tautologies were used, ranging in complexity from basic to complicated. The experimental evaluation uses a number of topologies to demonstrate how well the suggested method performs. The part that follows will display each topology with a selected route.

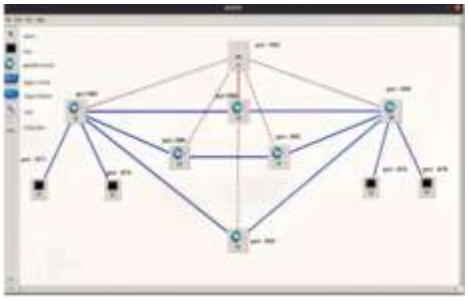


Figure3. Topology 6 nodes

First, we must determine the quantity and position of nodes in order to obtain a software-defined network. A network is constructed using Mininet design with the controller, four hosts, and six switches, as seen in

Fig. 3.Routed traffic using the cuckoo algorithm with restrictions on path selection.

Table 2: Statistics Ping between host1 and host4

When we send 5 packets only from 192.168.1.1 to 192.168.1.4 _Topology 6						
Packet	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.1	192.168.1.4	ICMP	98	
2	0.000065168	192.168.1.4	192.168.1.1	ICMP	98	
3	0.795141023	192.168.1.1	192.168.1.4	ICMP	98	
4	0.795175883	192.168.1.4	192.168.1.1	ICMP	98	
5	1.794367417	192.168.1.1	192.168.1.4	ICMP	98	
6	1.794387027	192.168.1.4	192.168.1.1	ICMP	98	
7	2.819737783	192.168.1.1	192.168.1.4	ICMP	98	
8	2.819787134	192.168.1.4	192.168.1.1	ICMP	98	
9	3.843120078	192.168.1.1	192.168.1.4	ICMP	98	
10	3.843237527	192.168.1.4	192.168.1.1	ICMP	98	

By sending a limited number of packets—only five—from the location to the source test the connection between the nodes, the information in table 2 above was obtained using the network monitor Wire Share.

Table 3: Best path (selection and discovery)

Path discovery	Hops	Path selection
h1→S1→S2→S6→h4	3	√
h1→S1→S3→S5→S6→h4	4	×
h1→S1→S4→S6→h4	3	√

Table 3 displays the topology's best route, which sends a packet from h1 to h4, where h is referred to as the host when choosing the best path.

We should also point out that we are using the ping command to verify the transmission and reception communication of info between the sender and the receiver. As shown in the accompanying table 4 and figure 4, by opening Wireshark on the other side to display the network analysis, we were also able to test the TCP connection by sending a number of packets over a period of 15 seconds.

Table 4: TCP Protocol analysis

Server listening on TCP port 5566			
[ID] Interval	Transfer	Bandwidth	
[6] 0.0- 1.0 sec	1.49 GBytes	12.8 Gbits/sec	
[6] 1.0- 2.0 sec	1.58 GBytes	13.6 Gbits/sec	
[6] 2.0- 3.0 sec	1.56 GBytes	13.4 Gbits/sec	
[6] 3.0- 4.0 sec	1.63 GBytes	14.0 Gbits/sec	
[6] 4.0- 5.0 sec	1.52 GBytes	13.0 Gbits/sec	
[6] 5.0- 6.0 sec	1.50 GBytes	12.9 Gbits/sec	
[6] 6.0- 7.0 sec	1.52 GBytes	13.0 Gbits/sec	
[6] 7.0- 8.0 sec	1.51 GBytes	13.0 Gbits/sec	
[6] 8.0- 9.0 sec	1.92 GBytes	16.5 Gbits/sec	
[6] 9.0-10.0 sec	1.78 GBytes	15.3 Gbits/sec	
[6] 10.0-11.0 sec	1.63 GBytes	14.0 Gbits/sec	
[6] 11.0-12.0 sec	1.60 GBytes	13.7 Gbits/sec	
[6] 12.0-13.0 sec	1.90 GBytes	16.3 Gbits/sec	
[6] 13.0-14.0 sec	1.74 GBytes	14.9 Gbits/sec	
[6] 14.0-15.0 sec	1.46 GBytes	12.6 Gbits/sec	
[6] 0.0-15.0 sec	24.4 GBytes	14.0 Gbits/sec	

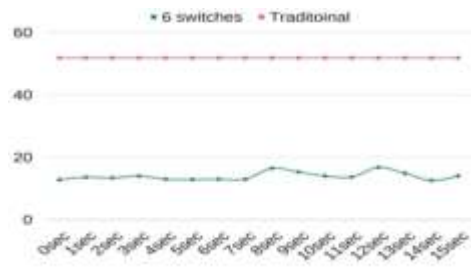


Figure4. TCP Protocol analysis through packets/15sec

With one control, we construct 12 switches and four hosts in this architecture. Fig. 5 shows the number of hosts and switches. If we want to transmit a message from host h1 to host h4, we must choose the optimum path from source to destination.

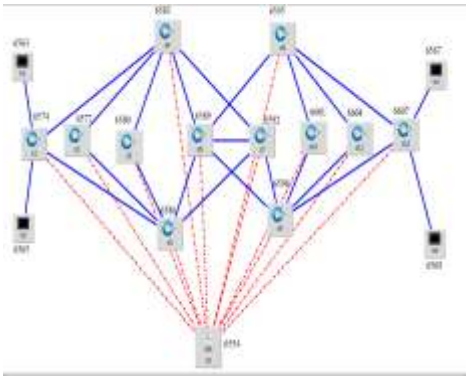


Figure5. Topology 12 Nodes

Table 5 depicts the best path from the source to the target.

Table 5: Simple best path (selection and discovery)

Path discovery	Hops	Path selection
h1→S1→S5→S3→S4→S6→S7→S9→S10→S8→S12→h4	5	√
h4→S12→S9→S6→S4→S1→h1	5	√

Table 6: Statistics Ping between host1 and host4

When we send 5 packets only from 192.168.1.1 to 192.168.1.4 _Topology 12:						
Packet	Time	Source	Destination	Protocol	Length	Info
1.	0.000000000	192.168.1.1	192.168.1.4	ICMP	98	
2.	0.000092810	192.168.1.4	192.168.1.1	ICMP	98	
3.	0.781305977	192.168.1.1	192.168.1.4	ICMP	98	
4.	0.781410810	192.168.1.4	192.168.1.1	ICMP	98	
5.	1.782305931	192.168.1.1	192.168.1.4	ICMP	98	
6.	1.782391266	192.168.1.4	192.168.1.1	ICMP	98	
7.	2.78890712[illegible]	192.168.1.1	192.168.1.4	ICMP	98	
8.	2.788981730	192.168.1.4	192.168.1.1	ICMP	98	
9.	3.811508913	192.168.1.1	192.168.1.4	ICMP	98	
10.	3.811558816	192.168.1.4	192.168.1.1	ICMP	98	

Table 7: TCP Protocol analysis

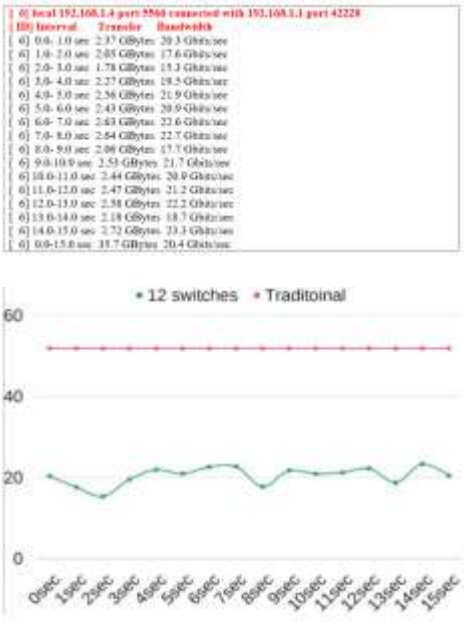


Figure6. Protocol analysis through packets/15sec

With one control, we generate 15 switches and four hosts. There are various paths to transmit a message from host h1 to host h3, however the optimum path for the sender may be h1, S1, S3, S5, S8, S13, S15, h4. Also, we can respond to the sender by path h4S15S5S3S1h1 or another path, as shown in Fig. 7.

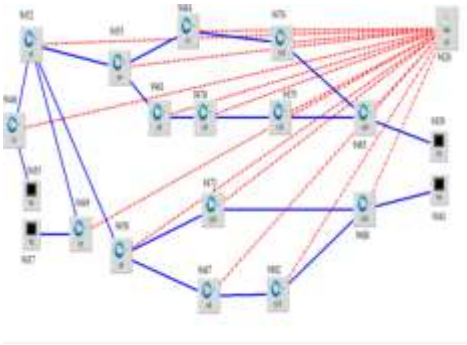


Figure7. Topology 15 nodes

Table 8: Statistics Ping between host1 and host

When we send 5 packets only from 192.168.1.1 to 192.168.1.4 _Topology 15:						
Packet	Time	Source	Destination	Protocol	Length	Info
1.	0.000000000	192.168.1.1	192.168.1.4	ICMP	98	
2.	0.000045851	192.168.1.4	192.168.1.1	ICMP	98	
3.	0.784220284	192.168.1.1	192.168.1.4	ICMP	98	
4.	0.784269885	192.168.1.4	192.168.1.1	ICMP	98	
5.	1.785483509	192.168.1.1	192.168.1.4	ICMP	98	
6.	1.785521773	192.168.1.4	192.168.1.1	ICMP	98	
7.	2.821947892	192.168.1.1	192.168.1.4	ICMP	98	
8.	2.821987841	192.168.1.4	192.168.1.1	ICMP	98	
9.	3.839085504	192.168.1.1	192.168.1.4	ICMP	98	
10.	3.839094468	192.168.1.4	192.168.1.1	ICMP	98	

Table 9: xterm result between h1 and h4 using TCP Protocol

[4] local 192.168.1.4 port 5566 connected with 192.168.1.1 port 42548		
[ID] Interval	Transfer	Bandwidth
[4] 0.0- 1.0 sec	1.37 GBytes	11.8 Gbits/sec
[4] 1.0- 2.0 sec	1.41 GBytes	12.1 Gbits/sec
[4] 2.0- 3.0 sec	1.33 GBytes	15.2 Gbits/sec
[4] 3.0- 4.0 sec	1.32 GBytes	11.4 Gbits/sec
[4] 4.0- 5.0 sec	1.44 GBytes	12.4 Gbits/sec
[4] 5.0- 6.0 sec	1.43 GBytes	12.4 Gbits/sec
[4] 6.0- 7.0 sec	1.45 GBytes	12.5 Gbits/sec
[4] 7.0- 8.0 sec	1.29 GBytes	11.1 Gbits/sec
[4] 8.0- 9.0 sec	1.35 GBytes	11.6 Gbits/sec
[4] 9.0-10.0 sec	1.30 GBytes	11.2 Gbits/sec
[4] 10.0-11.0 sec	1.28 GBytes	11.0 Gbits/sec
[4] 11.0-12.0 sec	1.29 GBytes	11.1 Gbits/sec
[4] 12.0-13.0 sec	1.23 GBytes	10.6 Gbits/sec
[4] 13.0-14.0 sec	1.32 GBytes	11.3 Gbits/sec
[4] 14.0-15.0 sec	1.44 GBytes	12.4 Gbits/sec

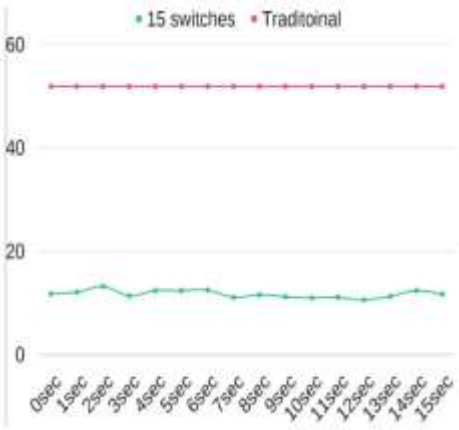


Figure8. TCP Protocol analysis through packets/15sec

Fig. 9 depicts 20 nodes used to construct a complex topology between the source and the distension. We used 20 switches and four hosts in this topology. An undirected graph connects the hosts and switches. One controller controls all of these nodes.

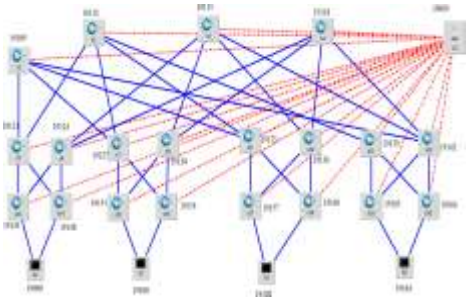


Figure9. Topology 20 nodes

Table 10: Simple best path (selection and discovery)

Path discovery	Costs	Path selection
h1.13.5.1.9.2.7.16.8.4.10.3.12.20.11.19. h4	5	√
h4.19.12.4.8.16.7.1.5.14.6.13. h1	5	√

Table 11: Ping result between host1 and host4

When we send 5 packets only from 192.168.1.1 to 192.168.1.4 ,Topology 12:						
Packet	Time	Source	Destination	Protocol	Length	Info
1.	[redacted]	192.168.1.1	192.168.1.4	ICMP	98	[redacted]
2.	[redacted]	192.168.1.4	192.168.1.1	ICMP	98	[redacted]
3.	[redacted]	192.168.1.1	192.168.1.4	ICMP	98	[redacted]
4.	[redacted]	192.168.1.4	192.168.1.1	ICMP	98	[redacted]
5.	[redacted]	192.168.1.1	192.168.1.4	ICMP	98	[redacted]
6.	[redacted]	192.168.1.4	192.168.1.1	ICMP	98	[redacted]
7.	[redacted]	192.168.1.1	192.168.1.4	ICMP	98	[redacted]
8.	[redacted]	192.168.1.4	192.168.1.1	ICMP	98	[redacted]
9.	[redacted]	192.168.1.1	192.168.1.4	ICMP	98	[redacted]
10.	[redacted]	192.168.1.4	192.168.1.1	ICMP	98	[redacted]

Table12: xterm result between h1 and h4 using TCP Protoco

Figure10. TCP Protocol analysis through packets/15sec

Fig. 11 depicts the largest architecture, with 30 nodes and two hosts. Assume that these nodes represent the topology of a country, a large city, or even cloud computing, and that we want to choose the optimum path from the transmitter to the recipient.

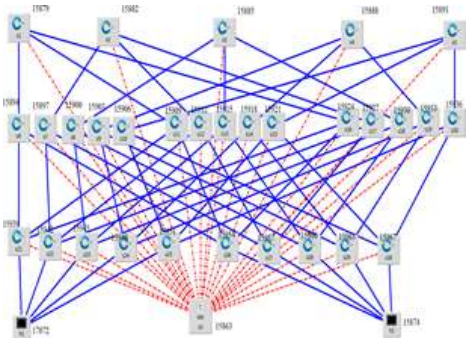


Figure11. Topology 30 nodes

Table13: Simple best path (selection and discovery)

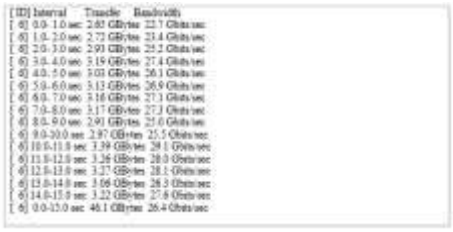
Path discovery	cost	Path selection
h1, s21, s16, s26, h2	3	√
h2, s25, s11, s1, s16, s21, h1	3	√

Table14: Ping result between host1 and host4

When we send 1 packets only from 192.168.1.1 to 192.168.1.4 _Topology 10:

Packet	Time	Source	Destination	Protocol	Length	Info
1.	0.000000000	192.168.1.1	192.168.1.4	ICMP	98	
2.	0.000048000	192.168.1.4	192.168.1.1	ICMP	98	
3.	0.788240271	192.168.1.1	192.168.1.4	ICMP	98	
4.	0.788278222	192.168.1.4	192.168.1.1	ICMP	98	
5.	1.788067338	192.168.1.1	192.168.1.4	ICMP	98	
6.	1.788114074	192.168.1.4	192.168.1.1	ICMP	98	
7.	2.821178045	192.168.1.1	192.168.1.4	ICMP	98	
8.	2.821219226	192.168.1.4	192.168.1.1	ICMP	98	
9.	3.845842988	192.168.1.1	192.168.1.4	ICMP	98	
10.	3.845782858	192.168.1.4	192.168.1.1	ICMP	98	

Table15: xterm result between h1 and h4 using TCP Protoco



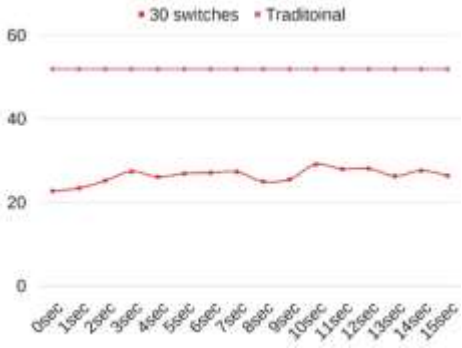


Figure12. TCP Protocol analysis through packets/15sec

C. Comparation Between SDN and Traditional

The information in Table 16 below shows the differences between traditional networking methods and software-defined networking methods, as well as the various kinds of software-defined networks and the timing of data delivery when using the TCP protocol. As depicted in the accompanying figures, milliseconds.

Table 16: Average Round-Trip Time between nodes x and y

Average round trip time (rtt) between node h1 and h16		
	the traditional network	OpenFlow-enabled Network
Single Topology	7.2 ms	4.6 ms
Linear Tobology	57.6 ms	41.7 ms
Tree Tobology	14.4 ms	8.8 ms
CustomTobology	10.1 ms	8.07 ms

Fig. 13 illustrates the stark contrast in packet transmission rate and target access speed between software-defined networking technology and conventional networks.

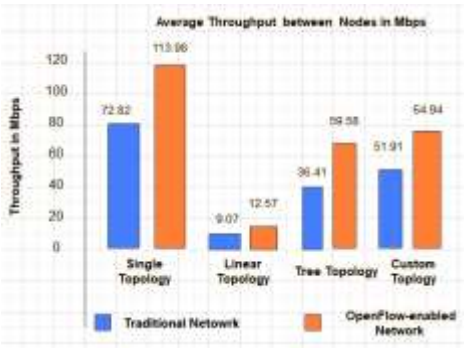


Figure13.Comparation between Traditional and SDN

In the image below, we can also see that the total of the network sizes discussed in the practical side of the message was able to transfer the same number of packets in the same amount of time, which is 15 seconds.

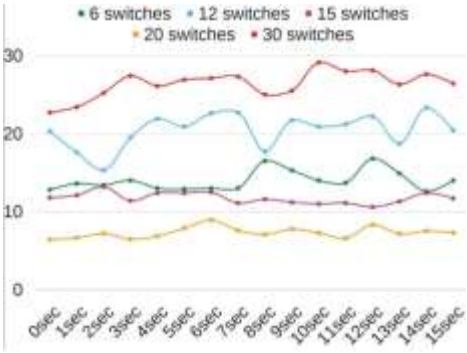


Figure14.TCP Protocol analysis through Packets/15sec

In another test, all of the models proposed in the practical side of the message were able to deliver the requested packages within the same timeframe, as seen in the picture below.

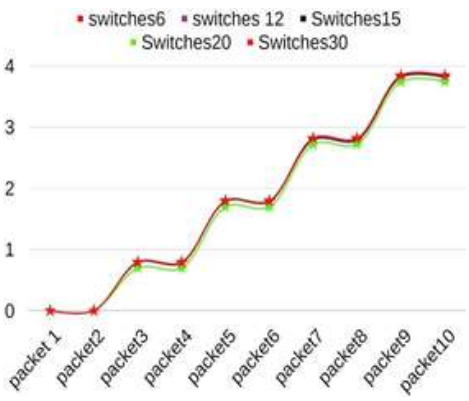


Figure15.Sending a packet of data in different types 6, 12, 15, 20 and 30 Nodes

6. CONCLUSION

This technology has proven its worth in centralizing the network and centralizing the numbers of all its components. We were able to build different types of networks of varying sizes programmatically using the Python language. We were able to find the best path in terms of time and cost. Using the network building algorithm, we were able to build different sizes of networks. We were able to use the Spanning Tree algorithm to get rid of the loop when building networks. Also, we used the algorithm to determine the price of the discovered paths.

Finally, use the cuckoo search algorithm to discover the best path among the detected paths based on the value of the fitness function. The physical fitness function includes two possibilities: either accepting the path to be the best or rejecting it. The results showed that all networks were able to send and receive data in a record time in milliseconds, much less than the time taken by traditional networks, and their performance was good depending on the data shown in the tables, sending packets in the fourth pure credit for each network. The results of the Wireshark also showed the process of sending packets and the problems that may occur when sending packets.

7. FUTURE WORK

In this thesis, various approaches to selecting the best path and dynamic load balancing are discussed. The offered techniques have been successfully used on a number of configurations. A summary of some of the recommended points is provided below.

- There was only one control in this research. In the future, though, multiple controllers may be employed.
- We used cuckoo search to choose the best route; alternative optimization methods include firefly, ACO, etc.

- In order to evaluate the efficacy of this approach, other layouts, such as 50 or 100, may be used.
- Try removing a connection and observe whether the network keeps operating or crashes.
- Make a backup copy of the controller to ensure that, in the event that the network's main controller malfunctions, the network will still operate.
- Responding to potential assaults on the central controller.
- Countering false information link injection assaults.
- Coming up with a strategy to lessen the load on the central controller in the event that the network is huge and rapid data delivery is necessary.
- Guard against distributed denial-of-service attacks on the network (DDoS).
- Deep packet inspection (IPS DPI) as part of an intrusion prevention system.

Acknowledgments

I'd like to take this chance to express my gratitude to my mentors for their guidance throughout the course of this endeavor. I also want to express my gratitude to my folks, who have always been my biggest supporters.

References

[1] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," IEEE/ACM Trans. Netw., vol. 14, no. 5, pp. 951–964, 2006, doi: 10.1109/TNET.2006.882843.

[2] S. Civanlar, E. Lokman, B. Kaytaz, and A. M. Tekalp, "Distributed management of service-enabled flow-paths across multiple SDN domains," 2015 Eur. Conf. Networks Commun. EuCNC 2015, pp. 360– 364, 2015, doi: 10.1109/EuCNC.2015.7194099.

[3] S. Islam, N. Muslim, and J. W. Atwood, "A Survey on Multicasting in Software-Defined Networking," IEEE Commun. Surv. Tutorials, vol. 20, no. 1, pp. 355–387, 2018, doi: 10.1109/COMST.2017.2776213.

[4] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software- defined networking (SDN): a survey," Secur. Commun. Networks, vol. 9, no. 18, pp. 5803–5833, 2016, doi: 10.1002/sec.1737.

[5] A. Wicaksana, "済無No Title No Title No Title," Https://Medium.Com/, pp. 1–26, 2016, [Online]. Available: https://medium.com/@arifwicaksanaa/pengertian-use-case-a7e576e1b6bf.

[6] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN)," Comput. Networks, vol. 112, pp. 279–293, 2017, doi: 10.1016/j.comnet.2016.11.017.

[7] M. R. Belgaum, S. Musa, M. M. Alam, and M. M. Su'Ud, "A Systema Review of Load Balancing Techniques in Software-Defined Networking," in IEEE Access, 2020, vol. 8, pp. 98612–98636. doi:10.1109/ACCESS.2020.2995849.

[8] W. Jiawei, Q. Xiuquan, and N. Guoshun, "Dynamic and adaptive multi- path routing algorithm based on software-defined network," Int. J. Distrib. Sens. Networks, vol. 14, no. 10, 2018, doi: 10.1177/155014771880568.

[9] Y. Aldwyan and R. O. Sinnott, "Latency-aware failover strategies for containerized web applications in distributed clouds," Futur. Gener. Comput.

Syst., vol. 101, pp. 1081–1095, 2019, doi: 10.1016/j.future.2019.07.032.

[10] H. H. Hsieh and K. Wang, "A Simulated Annealing-based Efficient Failover Mechanism for Hierarchical SDN Controllers," IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON, vol. 2019-Octob, pp. 1483–1488,2019, doi: 10.1109/TENCON.2019.8929249.

[11] D. Lopez-Pajares, J. Alvarez-Horcajo, E. Rojas, J. A. Carral, and I. Martinez-Yelmo, "One-Shot Multiple Disjoint Path Discovery Protocol (1S-MDP)," IEEE Commun. Lett., vol. 24, no. 8, pp. 1660–1663, 2020, doi: 10.1109/LCOMM.2020.2990885.

[12] H. K. Kakahama and M. Taha, "Adaptive Software-defined Network Controller for Multipath Routing based on Reduction of Time," UHD J. Sci. Technol., vol. 4, no. 2, pp. 107–116, 2020, doi: 10.21928/uhdjst.v4n2y2020.pp107-116.

[13] K. Kirkpatrick, "Software-Defined Networking," Commun. ACM, vol. 56, no. 9, pp. 16–19, 2013, doi: 10.1145/2500468.2500473.

[14] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A Survey on Software-Defined Networking," IEEE Commun. Surv. Tutorials, vol. 17, no. 1, pp. 27–51, 2015, doi: 10.1109/COMST.2014.2330903.

[15] G. Altangerel, T. Chuluuntsetseg, and D. Yamkhin, "Performance analysis of SDN controllers: POX, Floodlight and Opendaylight," no. Ifost, 2021, [Online]. Available: http://arxiv.org/abs/2112.10387

[16] D. F. Macedo, D. Guedes, L. F. M. Vieira, M. A. M. Vieira, and M. Nogueira, "Programmable networks-From software-defined radio to software-defined networking," IEEE Commun. Surv. Tutorials, vol. 17, no. 2, pp. 1102–1125, 2015, doi: 10.1109/COMST.2015.2402617.

[17] N. McKeown, T. Anderson, and H. Balakrishnan, "OpenFlow:white paper enabling innovation in campus networks," Acm Sigcomm, vol. 38, no. 2, pp. 69–74, 2008, [Online]. Available: http://dl.acm.org/citation.cfm?id=1355746

[18] M. Choi, H. Choi, and J. W. Hong, "XM L- Ba sed CO n f i g u ration M a nag em en t for IP Network Devices," IEEE Commun. Mag., no. July, pp. 84–91, 2004.

[19] V. Antonenko and R. Smelyanskiy, "Global network modelling based on mininet approach," HotSDN 2013 - Proc. 2013 ACM SIGCOMM Work. Hot Top. Softw. Defin. Netw., pp. 145–146, 2013, doi: 10.1145/2491185.2491211.

[20] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," Conex. 2012 - Proc. 2012 ACM Conf. Emerg. Netw. Exp. Technol., pp. 253–264, 2012, doi: 10.1145/2413176.2413206.

[21] B. Lantz and B. O'Connor, "A Mininet-based Virtual Testbed for Distributed SDN Development," Comput. Commun. Rev., vol. 45, no. 4, pp. 365–366, 2015, doi: 10.1145/2785956.2790030.

[22] D. Syrivelis et al., "Pursuing a software defined information-centric network," Proc. - Eur. Work. Softw. Defin. Networks, EWSDN 2012, pp. 103–108, 2012, doi: 10.1109/EWSDN.2012.20.

[23] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," Proc. 9th ACM Work. Hot Top. Networks, Hotnets-9, pp. 1–6, 2010, doi: 10.1145/1868447.1868466.

[24] A. Frömmgen, D. Stohr, J. Fornoff, W. Effelsberg, and A. Buchmann, "Capture and Replay," pp. 621–622, 2016, doi: 10.1145/2934872.2959076.

[25] D. Kumar and M. Sood, "Software Defined Networks (S.D.N): Experimentation

with Mininet Topologies," Indian J. Sci. Technol., vol. 9, no. 32, 2016, doi: 10.17485/ijst/2016/v9i32/100195.

[26] "Weerawardhana, J. L. M. N., Chandimal, N. J. A. W., & Bandaranayake, A. (2015). SDN testbed for undergraduate education. In Proceedings of the Fourth Engineering Students' Conference at Peradeniya (ESCaPe'15) .," p. 2015, 2015

[27] H. Kim, J. Kim, and Y. B. Ko, "Developing a cost-effective OpenFlow testbed for small-scale Software Defined Networking," Int. Conf. Adv. Commun. Technol. ICACT, pp. 758–761, 2014, doi: 10.1109/ICACT.2014.6779064.