Open Access **Baghdad Science Journal** **P-ISSN: 2078-8665**
Published Online First: December 2020 **E-ISSN: 2411-7986**

DOI: http://dx.doi.org/10.21123/bsj.2021.18.1.0163

Smart Flow Steering Agent for End-to-End Delay Improvement in Software-Defined Networks

***Omar F. Hussain*[1*]** ***Bilal R. Al-Kaseem*[2]** ***Omar Z. Akif*[3]**

[1] Quality Assurance and Academic Performance Department, University of Baghdad, Baghdad, Iraq
[2] Department of Computer Engineering, College of Engineering, Al-Iraqia University, Baghdad, Iraq
[3] Department of Computer, College of Education for Pure Science (Ibn al-Haitham), University of Baghdad, Baghdad, Iraq
[*]Corresponding author: omar_alberqdar@uobaghdad.edu.iq[*] , bilal.al-kaseem@aliraqia.edu.iq , omar.z.a@ihcoedu.uobaghdad.edu.iq
[*]ORCID ID: https://orcid.org/0000-0002-2638-5135[*] , https://orcid.org/0000-0001-8264-6339 , https://orcid.org/0000-0003-1773-103X

Received 21/10/2019, Accepted 21/1/2020, Published Online First 6/12/2020, Published 1/3/2021

Abstract

To ensure fault tolerance and distributed management, distributed protocols are employed as one of the major architectural concepts underlying the Internet. However, inefficiency, instability and fragility could be potentially overcome with the help of the novel networking architecture called software-defined networking (SDN). The main property of this architecture is the separation of the control and data planes. To reduce congestion and thus improve latency and throughput, there must be homogeneous distribution of the traffic load over the different network paths. This paper presents a smart flow steering agent (SFSA) for data flow routing based on current network conditions. To enhance throughput and minimize latency, the SFSA distributes network traffic to suitable paths, in addition to supervising link and path loads. A scenario with a minimum spanning tree (MST) routing algorithm and another with open shortest path first (OSPF) routing algorithms were employed to assess the SFSA. By comparison, to these two routing algorithms, the suggested SFSA strategy determined a reduction of 2% in packets dropped ratio (PDR), a reduction of 15-45% in end-to-end delay according to the traffic produced, as well as a reduction of 23% in round trip time (RTT). The Mininet emulator and POX controller were employed to conduct the simulation. Another advantage of the SFSA over the MST and OSPF is that its implementation and recovery time do not exhibit fluctuations. The smart flow steering agent will open a new horizon for deploying new smart agents in SDN that enhance network programmability and management.

Key words: E2E Delay, SDN, Smart Agent, Traffic Steering.

Introduction:

The network landscape has undergone unprecedented transformations in recent times. The shift from static networks to mobile and dynamic networks has been catalyzed by a variety of innovations, including the introduction of smart devices, mobility, wireless access, virtualization and the cloud (1). An increase in large data and network traffic, as well as novel categories of connected devices (e.g. industrial machines, thermostats, sensors, actuators, smart vehicles, wearable devices, and smart appliances), can all be supported by mobile networks thanks to the development of the Internet-of-Things (IoT) (2,3).

The number of connected devices is 9 billion at the moment and, by 2020, it is estimated to reach 24 billion (4). However, problems of element control and excessive network loading have already started to confront carriers (5,6). This may lead to network breakdown if adequate measures are not taken to deal with the influx of IoT in which the things are not merely network consumers, but also traffic generators (2). The processing, analysis and secure storage of the terabytes of data produced by smart IoT devices must be ensured (7).

To be able to keep up with the dynamic requirements of network bandwidth and computational power, networks must evolve better intelligence, speed, security, reliability and scalability. In earlier times, gateways and routers executed proprietary applications on proprietary

hardware satisfied the needs of static networks. However, since programmable networks have substituted static networks, those gateways and routers are unable to handle the issues posed by dynamic networks. Under these circumstances, service providers do not only deal with the network requirements and manage the high number of connected devices, but also to maintain their competitiveness should explore alternatives like software-defined networking (SDN). Network and bandwidth problems can be effectively solved by SDN by dividing the control and data planes and therefore depriving the forwarding function of control and promoting dynamic networks that can be centrally programmed (8).

Transmission of information as digital packets at global level is made possible by the distributed control and transport network protocols contained in routers and switches. Standard IP networks display great complexity and their management is challenging, even though they are used on a broad scale (9). Configuration of every network device has to be undertaken separately by network operators with low-level commands that also frequently differ between vendors, in order to apply the targeted high-level network policies. Aside from the complicated configuration, fault dynamics and load modifications also burden network environments. Existing networks display vertical integration and most IP networks lack automatic reconfiguration and response mechanisms. The difficulty of making the networking infrastructure more flexible and innovative stems from the incorporation of the control and data planes, respectively responsible for network traffic decisions and traffic forwarding based on prior decision-making, into the networking devices (10,11).

Figure 1 presents the new networking paradigm that shows great potential in enhancing the weaknesses of existing network infrastructures, the new infrastructure is called software-defined networking (SDN) (12)(13). It works by separating the control and data planes, as a result of which the network is no longer vertically integrated. This division facilitates policy implementation and network reconfiguration and innovation, since network switches become mere forwarding devices and the control logic is applied in a controller with logical centralization (14). SDN was developed to increase network programmability and management through the centralized controller.

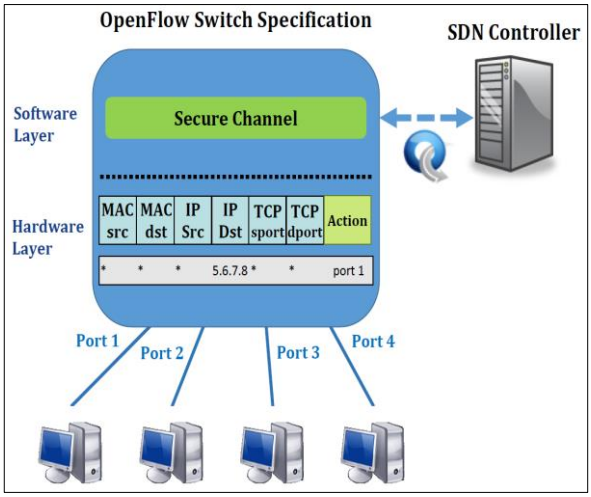


Figure 1. The SDN architecture

However, this solution is hindered by the necessity to ensure that the networks perform well and are scalable and reliable. Consequently, physical distribution of control planes is undertaken in production-level SDN network designs (15,16). The introduction of a clearly defined software routine between SDN-based switches and the controller can help to disassociate the network planes (data and control). OpenFlow is a well-known application program interface (API) (17,18) in SDN. This interface empowers the control plane to straightway handle the status of the data plane components.

At least one table of packet managing rules, known as a flow table, is contained in an OpenFlow switch and every rule corresponds to a traffic subset on which it conducts specific tasks, like dropping, forwarding, editing, etc. A controller application can order an OpenFlow switch to adopt the behavior of a router, firewall, or other components, including load balancer, traffic shaper and middle box components, according to the rules applied by that controller. The division of the aspects of network policy formulation, application of network policies in switching hardware and traffic forwarding is a key implication of the SDN concepts. Without such division, it would be impossible to achieve the wanted flexibility, separate the issue of network control into smaller, manageable issues, as well as improving network management and enabling network development through facilitation of the creation and implementation of novel abstractions in networking.

An enhanced routing algorithm for the SDN environment is put forth in this study. To this end, a simulation is conducted with the POX controller serving as an open source SDN controller. Equipped with SDN Python libraries and APIS, the POX controller determines the shortest path and the source-to-destination paths with the Open Shortest

Path First (OSPF) algorithm and the Minimum Spanning Tree (MST) algorithm, respectively. However, whereas the shortest path with minimal number of hops is the target of the OSPF and MST applied in the POX controller, the ideal path with least number of hops and minimal delay is assessed in the suggested scheme, as the shortest path may be unsuitable if the link delay among connected nodes in a network is uneven. Hence, the POX controller is reconfigured in this study to be able to determine the best path, which is not the path with this least number of hops but the path with that least delay, based on this information about the delay of every network link.

The structure of the remainder of the paper consists of six further parts. In Section 2, relevant existing research is reviewed, while Section 3 addresses proactive and reactive flow tables, and Section 4 focuses on the traffic engineering and routing mechanisms. In Section 5, the suggested scheme is presented and, in Section 6, the outcomes of the performance assessment are provided. The study conclusion is given in Section 7.

Related Works

In the following part, research pertinent to the incorporation of algorithms for path identification in the SDN controller is reviewed. Additionally, the existing opportunities for integrating these algorithms with the OpenFlow protocol in the SDN framework are explored.

Hindering occurrence of congestion or regulating the traffic in such a way as to minimize the implications of congestion has been proposed by numerous researchers as a strategy for improving load balancing. Unlike other congestion-aware protocols that took into account just the local status because of the problems of path congestion measurement, one study put forth a new congestion-aware routing protocol capable of making routing decisions based on both local and remote congestion information (8). The congestion is innovatively represented for links between routers and an aggregation protocol facilitated the dissemination of the congestion statistics to other network routers. Hence, both local and potential next-hop status could be considered in routing decision-making thanks to such an effective approach amenable to scalability.

A different routing strategy known as Accumulative Load-Aware Routing (ALAR) for SDN and considering the aggregating flow load on every link was suggested in (17). This strategy improved network resource use and reduced congestion likelihood, and implicitly, average end-to-end delay, by establishing routes based on information regarding the entire network topology and traffic.

To enhance certain parameters of network performance (e.g. link congestion and bandwidth usage) and make the conventional OSPF routing protocol perform better, one study endowed the OSPF of the SDN with new attributes (14). According to the outcomes of the simulation, besides identifying the site where congestion occurred or is likely to occur, a network with a smart dynamic controller could also employ an intelligent heuristic Flows Distribution Algorithm (FDA) to avoid congestion through effective management of chosen flows on chosen network routers.

In a different study, a user-based strategy of traffic improvement was applied, whereby users' application trends documented via generic NetFlow records were used to characterize those users to improve understanding of individual user usage (15). Furthermore, Linux-based hierarchical token bucket (HTB) queues with real-time adjustment to each user profile depending on priorities identified by the users was used to implement an SDN application for supervising and managing traffic. Thanks to this approach, packet loss and latency were significantly reduced for chosen high-priority users as dynamic distribution of bandwidth according to users' profile priority ensured the efficiency of the traffic management strategy under upstream as well as downstream network congestion.

Problem Statement and Contributions

The overall goal of the present study is to decrease the network delay through the introduction of a smart agent in the SDN controller, and thus to contribute to literature expansion, as not many studies have used optimized routing algorithms in SDN controllers. Additionally, validation of proof-of-concept application, which is concerned with methods of using the SDN controller in surveillance of extensive networks and diminishing end-to-end delay, is also an important objective of the study. In particular, the contributions of this study are significant for the following reasons:

1. Prevention and anticipation of congestion between a source and destination and delay reduction through the creation of a delay- and congestion-aware routing algorithm based on an open-source SDN controller;
2. Enhancement of network performance and throughput with optimized flow-table entries underpinned by link quality among OpenFlow switches;

3. Removal of the number of packets returned to the SDN controller for flow-table entry update and speeding up recovery in the event of failure through the incorporation of a smart agent in the SDN controller.

The suggested scheme is based on the smart flow steering agent (SFSA), which enables both routing algorithms and agent applications to be deployed in a dynamic and scalable manner in the SDN controller. Employing a multi-objective routing algorithm, the SFSA scheme is geared towards identifying the best path, thus improving SDN network management and configuration.

Reactive and Proactive Flow-Table in SDN

Dynamic restoration of flows is enabled by the flexibility exhibited by the SDN network configuration. This procedure involves performance of flow tasks, such as addition or elimination of flow-entries to/from flow-tables, by network devices to achieve disrupted flow re-routing (19). Upon reception of a packet, an attempt is made by the OpenFlow switch to establish a correspondence between the packet header and its flow table. The switch forwards the header to its OpenFlow controller if it cannot find any packet information. This results in a packet in event in the controller. The headers L2, L3 and L4 are assessed by the controller and the packet is distributed all through the network along with other packets of the same flow. The controller returns flow modification to the switches when it secures a route, so that the switches can update their flow tables as depicts in **Fig. 2** in which the proactive or reactive techniques are employed to create such data paths.

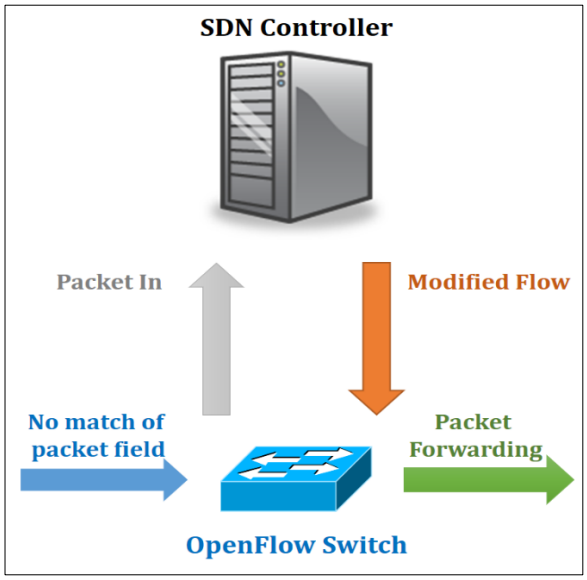


Figure 2. Reactive and proactive flow-table in SDN

Ensuring that packet transmission occurs after paths are established is the main task of proactive routing techniques in SDN networks. What makes these techniques particularly useful is that, in the process of end-host detection, they do not produce extra network overhead and do not interfere with network performance. Since forwarding rules are already in place, the controller does not receive any packet transmitted through the network. On the downside, prior mapping of every interconnection in the network topology is a requirement in proactive techniques and this may not be possible in some networks. In contrast, in cases where a switch receives a new packet that has no correspondent in its flow table, reactive routing techniques are applicable. The packet is sent to the controller, which undertakes identification of its end-hosts with control messages and, prior to forwarding the data traffic in the network, it acquires the active links. Reactive routing techniques are advantageous because they do not require prior information about the entire network topology. On the downside, given that the controller receives every initial packet of every new flow prior to network dissemination, the performance of extensive networks may be adversely affected by the volume of control messages.

Path establishment in SDN networks can be effectively undertaken with both proactive and reactive techniques. Path-deploying controllers must be synchronized in the case of proactive techniques, while extra overhead is generated by reactive techniques in extensive networks and they also affect network performance. Furthermore, owing to the caching of the next flow forwarding state on the OpenFlow switch, latency is introduced by reactive techniques only to the initial packet of flow (16).

Traffic Engineering and Routing Mechanisms

Improvement of network performance and traffic delivery relies greatly on traffic engineering, which in turn depends on route optimization, involving discovery of routes that could help attain the target network performance (20,21).

Important elements of traffic engineering in IP networks are quality of service (QoS) and resilience mechanisms. Alongside bandwidth requirements, QoS assurance (i.e. end-to-end delay, jitter, likelihood of packet loss, and energy efficiency) is also important in many of the novel multimedia applications. Meanwhile, IP networks are often affected by various kinds of failures, including node or link failure, and rapid resilience mechanisms are needed to address those failures (19,22,23). Under such circumstances, preventing

failures from disrupting network performance and resource usage is the main objective of traffic engineering solutions. Basic routing models consisting of shortest path and load-balancing strategies with traffic divided evenly into a specific number of paths of equal cost represent the cornerstone of the majority of IP-based traffic engineering solutions (23). In the following part, several popular routing algorithms and their impact on network traffic performance are discussed.

The connected graph $G(V,E)$ can be derived from the SDN topology, the generated graph is weighted and directed, where V is the vertices (SDN switches) and E is the connected edges of the graph (connected links).

Open Shortest Path First (OSPF)

An OpenFlow interface is included in numerous new Internet routers, which are also compatible with a hybrid OpenFlow/OSPF mode. For optimal forwarding of packets, the distributed legacy routing protocol is usually applied by hybrid control plane frameworks and is augmented with high-priority rules by the SDN controller to allow more elaborate routing configurations (24).

In the case of IP networks, the OSPF algorithm frequently serves as an interior gateway protocol (IGP) based on the link state. A network topology graph is created by OSPF with information about link state derived from existing routers. Furthermore, it applies a hop-counts technique to determine the shortest path for packet routing.

In the connected graph $G(V,E)$, the links weights are given by the weight function $w: E \rightarrow [0,\infty]$, the cost of the link can be expressed as the number of hops between two switches. Therefore $w(u,v)$ is the non-negative cost of moving directly from $u \in V$ to $v \in V$ with a cost of $d(v)$.

The shortest path algorithm can be executed as follows from source node s to v:

1. Set S to empty, where S is a set of nodes whose shortest paths from the source have already been determined;
2. Add the source s to S and $d(s) = 0$, if there is a link from s to v, $d(v) = w(s,v)$, for all other nodes, $d(v) = \infty$;
3. Add node u to s, where $d(u)$ is the minimum hops in $V - S$, if $S = V$, complete the task;
4. If there is a link from u to $v \in V - \$$; ; $\min\{d(v), d(u) + w(u,v)\}$. Then go back to step 3.

Minimum Spanning Tree (MST)

To prevent the issue of packet broadcast storm in a legacy network with loops, a spanning tree is created with the IEEE 802.11d distributed spanning tree protocol. The same issue can be addressed in an SDN network through central creation of a spanning tree based on the overall information of network topology gathered by the SDN controller (14). The basic bridge protocol that was initially devised for loop prevention in bridge network is known as the Spanning Tree Protocol. Data are disseminated through the tree constructed by this protocol and encompassing all network bridges, with only the links included in the tree being allowed. If the bridged network breaks down, it can repair itself through activation of the links that have been previously blocked. However, the algorithm presents limitations in that recovery in the event of failure does not occur fast enough and backup links can only be used for forwarding under conditions of failure (20).

The spanning tree T of the graph $G(V,E)$ is the graph $T = (V,E')$ such that T is a tree and $E' \subseteq E$ if G is a weighted graph, T will be a weighted graph. The total edge weight of T is defined as the sum of the edge weight in G. the minimum spanning tree T of G, is the spanning tree of G, such that no other spanning tree of G has lower total edge weight.

Proposed Smart Traffic Steering

Quality deterioration is mainly caused by network congestion, which occurs when the traffic is not distributed homogeneously or network resources are insufficient. This issue is compounded by the fact that current routing algorithms are designed to detect just the shortest paths from a source to a destination, and that routing and congestion control do not work together (25). Packet drop ratio (PDR), latency (end-to-end delay) and error rate are all heightened by the rise in queuing delay due to congestion.

Building on the extended Dijkstra's algorithm, the approach put forth in this study aims to identify the best path from a source to a destination in an SDN. The information collected by the OpenFlow switches is used by the SDN controller to create a general network topology. The suggested approach involves appraisal of congestion rate during network traffic by the OpenFlow switches through gauging of link delay via hello and acknowledgement messages for detection of connected switches. The detection information sent by every switch to the SDN controller enables the latter to construct a general network topology. Flow table entries are generated by an agent application with a smart flow steering mechanism working within the SDN controller. Subsequently, the flow tables are sent to the

Baghdad Science Journal
P-ISSN: 2078-8665
E-ISSN: 2411-7986

OpenFlow switches. Applying a technique underpinned by Dijkstra's algorithm, the smart flow steering mechanism determines the shortest path for every route, thus achieving packet routing. Furthermore, weights are allocated by the SFSA to all network links in order to identify the shortest paths. The allocation of link weights takes the form of link states, with the SDN controller receiving link state messages from all OpenFlow switches. For routing to be considered optimal, it must prevent network congestion by employing existing network resources for effective traffic distribution. To take into account both the edge weights and the switch congestion for all routes obtained from the underpinning SDN topology, the modified Dijkstra's shortest path algorithm is applied. Dijkstra's algorithm is classified as a label-setting algorithm because the label of a node that has been visited remains permanently fixed if it does not become out of reach. All nodes in label-setting algorithms have such a label, which informs about the cost and route for reaching a specific node from a source. The link delay and estimated congestion give the route cost function, which enables the SFSA to identify every potential path between source and destination. By contrast to the OSPF and MST algorithms, the route hop count and minimum segment are respectively disregarded in label-setting algorithms, which choose the route with least delay and congestion from among all potential routes. In SDN, the SFSA can achieve significant improvement in PDR and end-to-end delay. **Fig. 3** presents the SFSA framework.

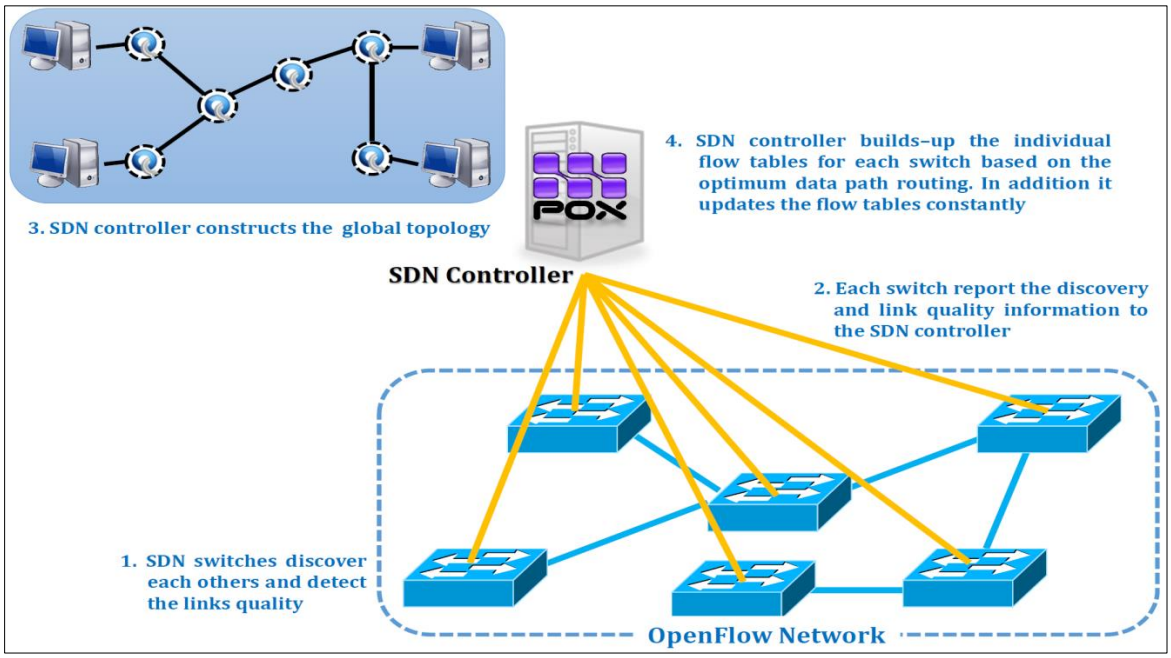


Figure 3.The proposed architecture of smart flow steering agent

At the controller, the global SDN topology is constructed based on the connected graph $G(V,E)$ introduced by Huang (26), in which the SDN network consists of N switches denoted by $\{V\}_{i=1}^{N}$ connected through link sets $E = e_{i,j}$ where $e_{i,j}$ is the link incident from node v_i to v_j. On the controller, P_j is defined

$$P_j = [L_1, L_2 \cdots L_N] \quad \ldots \quad (1)$$

As a set of N links from path j, and let $Cong_i$ is the congestion level of link L_i, the congestion level is being estimated at the switch using the following formula:

$$Congs.Switch = \frac{data\ rate}{bandwidth} \times Packet_{in} \quad \ldots \quad (2)$$

Where the $Packet_{in}$ is the of packets passed across the given switch. The congestion status (P_j^m) of path j is the maximum congestion level of all the links of path j:

$$P_j^m = max(congs_1, congs_2, \cdots, congs_N) \quad \ldots \quad (3)$$

Where $congs_i$ is the congestion level of link at switch i on the other hand, the congestion weight of the path P_j^m is the sum of all the links congestion level in path j:

$$P_j^w = sum(congs_1, congs_2, \cdots, congs_N) \quad \ldots \quad (4)$$

Once all the network information is being reported at the controller, the controller starts

building up the flow table entries for each switch, for a given source/destination pairs, let K be equal cost shortest paths where congestion level are $P_1^m, P_2^m, \cdots, P_k^m$ respectively. Among these paths, the controller defines a set S_x such that:

$$S_x = \{x | P_x^m = \min\{P_1^m, P_2^m, , P_K^m\}\} \quad \ldots (5)$$

$$S_y = \{y | P_y^w = \min\{P_x^w . x \in S_x\}\} \quad \ldots (6)$$

S_x contains all the paths that have the same minimum congestion level, S_y contains source destination pairs with lowest congestion weight. Finally, the paths is S_y is the Flow Table entities for each switch in the SDN controller.

Performance Evaluation Results

A Windows-based Oracle virtual box and MATLAB software for the Mininet emulator and algorithm analysis, respectively, were employed to perform the simulation scenarios. There are two sequential procedures undertaken by the SDN controller. Prior to selection of a routing algorithm, it conducts network discovery to uncover the network topology. The identification of the OpenFlow switches is followed by application of the SFSA in the SDN controller and provision of optimum flow tables to the OpenFlow switches for storage and initiation of data routing.

Two scenarios concerned with end-to-end delay and PDR are explored in this study via the following steps:

- Creation of the general network topology and execution of the default routing algorithm OSPF or MST in POX controller by the Mininet;
- Use of the packet capture method to extract link quality information from the OpenFlow switches following completion of the network discovery process;
- Comparison between the outcome of using SFSA in the SDN controller and the default algorithms based on MATLAB analysis;
- Re-simulation in MATLAB of the same network topology but with the link quality information derived from the Mininet;
- MATLAB-based application of three distinct routing algorithms in the same sub-domain, namely, OSPF and MST in custom as well as full-mesh topology, and SFSA underpinned by the extended Dijkstra's algorithm with parameters of link quality information (e.g. link delay, bandwidth utilisation, and number of hops).

Several assumptions were formulated before the two simulation scenarios and related analysis were initiated:

- The computation capacity and traffic handling are the same for every OpenFlow switch;
- The creation of the general topology does not include the connection between every OpenFlow switch and a remote SDN controller;
- The SDN switches and hosts are bi-directionally connected via SDN links;
- The simulation is conducted with the POX controller.

Mesh Topology Scenario

For both Mininet (**Fig. 4**) and MATLAB (**Fig. 5**), the mesh topology was made up of five OpenFlow switches with ten links ensuring their connection**. Fig. 6** illustrates the round-trip comparison and it can be seen that, by contrast to the default Mininet routing algorithms, the SFSA was associated with less delay by achieving a 23% decrease in packet round trip time (RTT) even when the generated packets across the network have been increased. On the other hand, **Fig. 7** presents the algorithm execution time and failure recovery time, indicating that, by comparison to the other algorithms, the SFSA can identify the optimal path from all possible paths much faster in the event of network failure.

Baghdad Science Journal
P-ISSN: 2078-8665
E-ISSN: 2411-7986

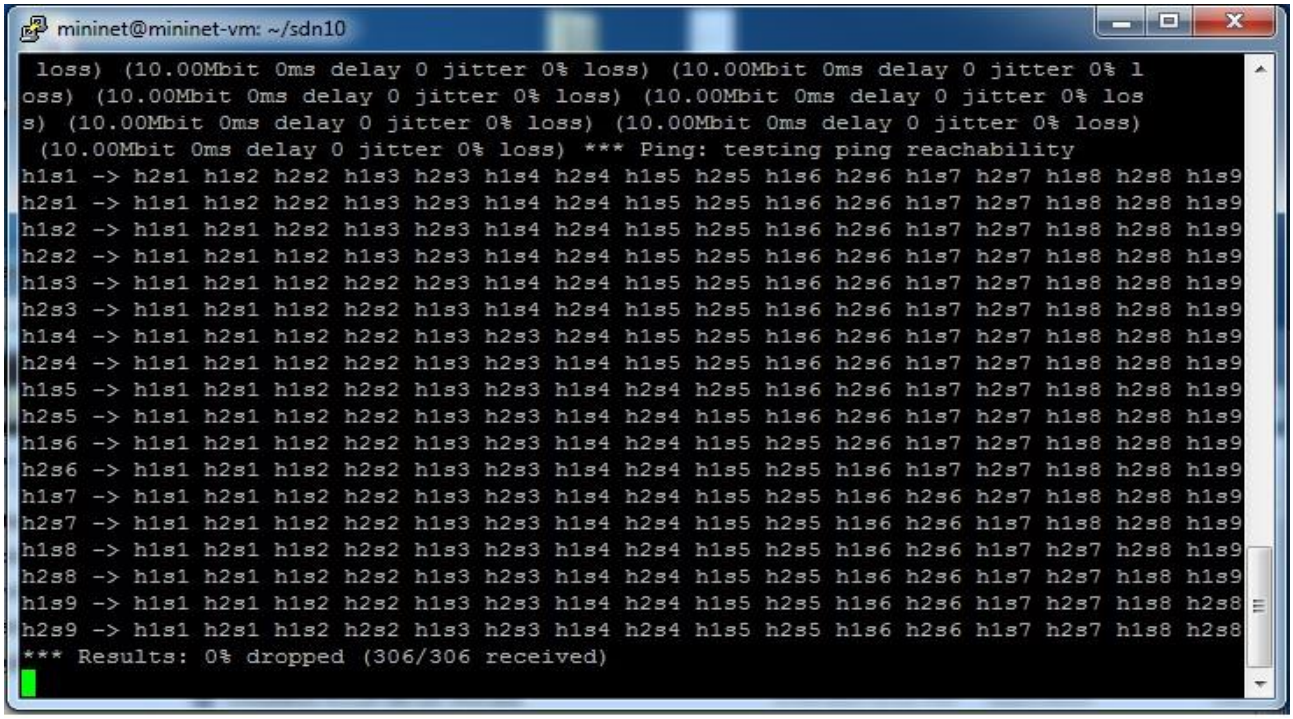


Figure 4. Mesh topology simulation run for the Mininet

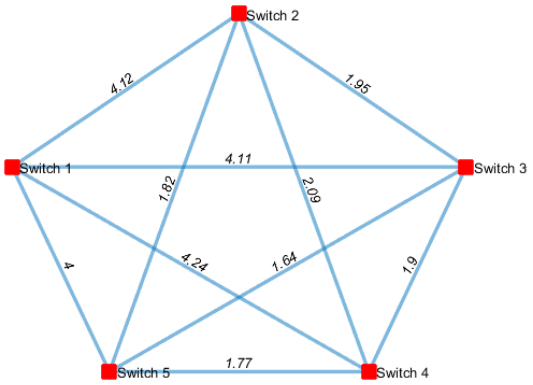


Figure 5. Mesh topology architecture using MATLAB

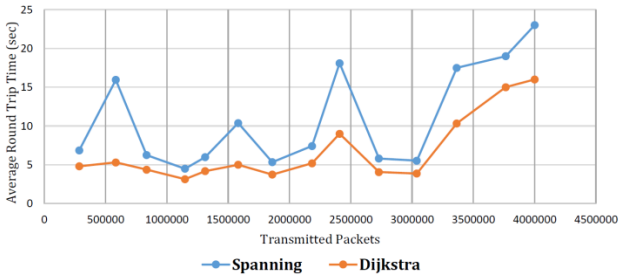


Figure 6. Mesh topology round-trip comparison

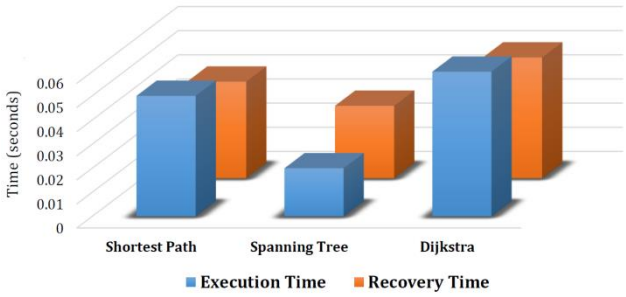


Figure 7. Mesh topology scenario for SFSA execution and recovery time

The values associated with the worst-case scenario in mesh topology are listed in Table I. It can be seen that, in the selection of the optimal path between two nodes, the SFSA successfully balanced path length and congestion cost. As a result, the end-to-end delay was 15-30% lower in the case of the SFSA than in the case of the default Mininet routing algorithm, which is influenced by the traffic load and node position.

Table 1. Mesh Topology Worst Case Scenario

Source	Destination	Hop-counts	Delay	Algorithm
1	4	1	4.24	OSPF
1	4	1	4.24	SFSA
1	4	2	5.77	MST
3	4	2	8.35	OSPF
3	4	2	3.14	SFSA
3	4	2	3.14	MST

Custom Topology Scenario

For both Mininet (**Fig. 8**) and MATLAB (**Fig. 9**), the custom topology was made up of nine OpenFlow switches with eleven links ensuring their connection. **Figure 10** presents the algorithm execution time and recovery time, indicating that, by comparison to the other algorithms, the SFSA can identify the optimal path from all possible paths much faster. **Figure 11** illustrates the PDR comparison and it can be seen that, by contrast to the default Mininet routing algorithms, MST and OSPF, the SFSA achieved a 2% decrease in PDR. The capacity of a control plane to prevent additional traffic loss in the event of a network failure by finding a feasible alternative routing with no routing loops and black holes is known as failure recovery.

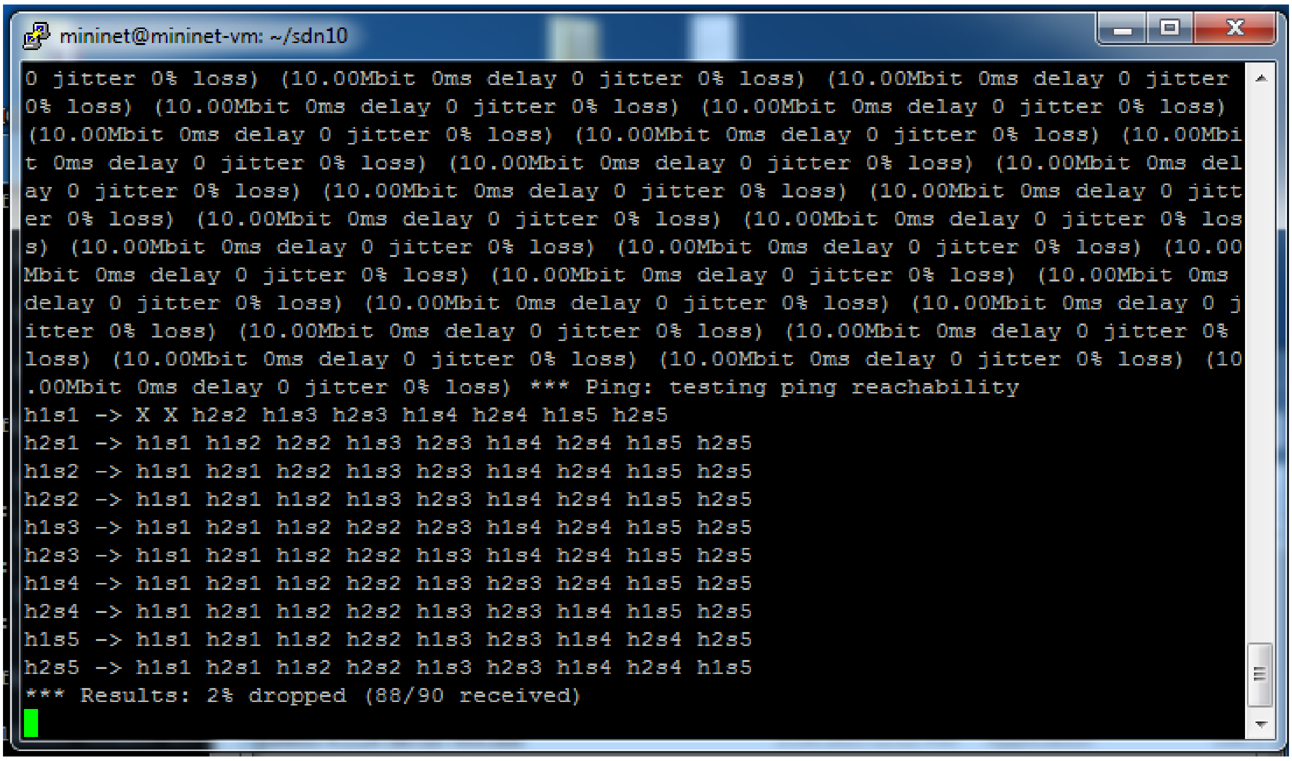


Figure 8. Custom topology simulation run for the Mininet

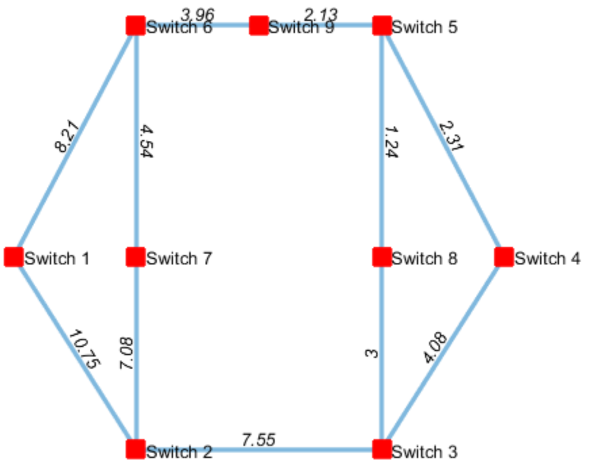


Figure 9. Custom topology architecture using MATLAB

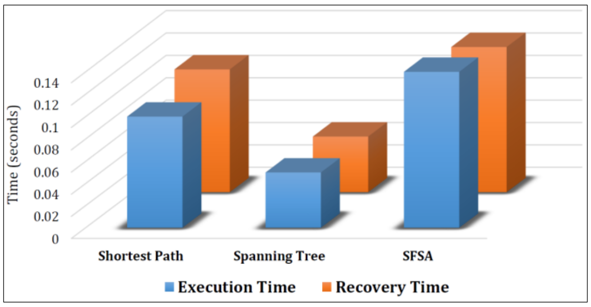
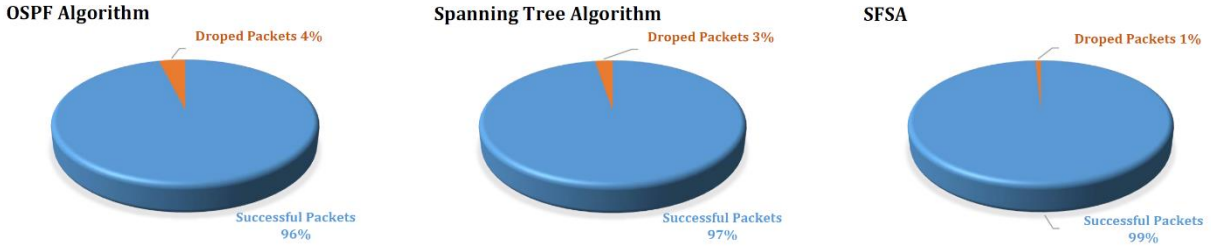


Figure 10: Custom topology scenario for SFSA execution and recovery time



(a) (b) (c)

Figure 11. Packet Drop Ratio (PDR %) for the custom topology scenario

In the process of optimal path identification, an important factor considered by the SFSA is the determination of the congestion of every network link. To achieve route optimization, it is not the shortest path that is chosen, but the path with minimal congestion and delay, and this selection process is undertaken with the multi-objective routing algorithm. The values associated with the worst-case scenario in custom topology are listed in Table 2. It can be seen that, in the selection of the optimal path between two nodes, the SFSA successfully balanced path length and congestion cost. As a result, the end-to-end delay was 15-45% lower in the case of the SFSA than in the case of the default Mininet routing algorithm, which is influenced by the traffic load of the switches.

Table 2. Custom Topology Worst Case Scenario

Source	Destination	Hop-counts	Delay	Algorithm
1	2	1	10.75	OSPF
1	2	1	10.75	SFSA
1	2	3	19.83	MST
3	4	3	22.38	OSPF
3	4	4	16.61	SFSA
3	4	4	16.61	MST

Conclusion:

This paper reviewed major research related to SDN, and proposed a promising approach involving incorporation of a smart agent in the SDN controller. SDN is of significant benefit to network and flow management, but it is not without limitations, which needs further investigation. One major problem is the stability of the control plane. Numerous strategies have been suggested to address this problem, but all of them involve measuring the performance of control plane stability in relation to certain parameters of network QoS (e.g. throughput and latency). In contrast to these strategies, the proposed SFSA approach is geared towards improvement of end-to-end delay, RTT and PDR. According to the findings of this study, as the number of requests rise at high traffic, the SFSA ensures the stability of the execution and recovery time in the two scenarios employed. In addition, by transferring knowledge to an external agent, it is possible to diminish the number of flow entries in the switches. The results of the simulation confirm that, by comparison to the MST and OSPF algorithms, the SFSA approach performs much better.

Authors' declaration:

- Conflicts of Interest: None.
- We hereby confirm that all the Figures and Tables in the manuscript are mine ours. Besides, the Figures and images, which are not mine ours, have been given the permission for re-publication attached with the manuscript.
- Ethical Clearance: The project was approved by the local ethical committee in University of Baghdad.

References:

1. Braun W, Menth M. Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices. Future Internet. 2014;6(2):302.
2. Gubbi J, Buyya R, Marusic S, Palaniswami M. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. FUTURE GENER COMP SY. 2013;29(7):1645 – 1660.
3. Al-Kaseem BR, Al-Dunainawi Y, Al-Raweshidy HS. End-to-End Delay Enhancement in 6LoWPAN Testbed Using Programmable Network Concepts. IEEE Internet of Things Journal (IOT-J). 2018 Nov 1;6(2):3070-86.
4. van der Meulen R. Analysts to Explore the Value and Impact of IoT on Business at Gartner Symposium / ITxpo 2015, November 8-12 in Barcelona, Spain;

2015. [Accessed on: Jun. 16, 2019]. http://www.gartner.com.
5. Chin WH, Fan Z, Haines R. Emerging technologies and research challenges for 5G wireless networks. IEEE Wireless Communications(IEEEWIRCOM). 2014 May 12;21(2):106-12.
6. Jara AJ, Zamora MA, Skarmeta A. Glowbal IP: An Adaptive and Transparent IPv6 Integration in the Internet of Things. Mob Inf Syst. 2012 Jul;8(3):177–197.
7. Akif OZ, Rodgers GJ, Al-Raweshidy HS. Protecting a Sensitive Dataset Using a Time Based Password in Big Data. In: 2017 Computing Conference; 2017. p. 871–879.
8. Al-Shabibi A, Martin B. MultiRoute - a Congestion-aware Multipath Routing Protocol. In: 2010 International Conference on High Performance Switching and Routing; 2010. p. 88–93.
9. Sabbeh A, Al-Dunainawi Y, Al-Raweshidy HS, Abbod MF. Performance Prediction of Software Defined Network Using an Artificial Neural Network. In: 2016 SAI Computing Conference (SAI); 2016. p. 80–84.
10. Sood K, Yu S, Xiang Y. Software-Defined Wireless Networking Opportunities and Challenges for Internet-of-Things: A Review. IEEE Internet of Things Journal (IOT-J). 2015 Sep 28;3(4):453-63.
11. Kreutz D, Ramos FMV, Veríssimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-Defined Networking: A Comprehensive Survey. Proceedings of the IEEE. 2015 Jan; 103(1):14–76.
12. Al-Kaseem BR, Al-Raweshidy HS. Enabling Wireless Software Defined Networking in Cloud Based Machine-to-Machine Gateway. In: 2016 8th Computer Science and Electronic Engineering (CEEC) (CEEC'16). Colchester, Essex, United Kingdom; 2016. p. 24–29.
13. He J, Song W. Achieving Near-Optimal Traffic Engineering in Hybrid Software Defined Networks. In: IFIP Networking Conference (IFIP Networking), 2015; 2015. p. 1–9.
14. Hasan H, Cosmas J, Zaharis Z, Lazaridis P, Khwandah S. Development of Performance of OSPF Network by Using SDN Concepts. In: Communications and Networking (BlackSeaCom), 2016 IEEE International Black Sea Conference on; 2016. p. 1–4.
15. Bakhshi T, Ghita B. User-Centric Traffic Optimization in Residential Software Defined Networks. In: 2016 23rd International Conference on Telecommunications (ICT); 2016. p. 1–6.
16. Jararweh Y, Al-Ayyoub M, Darabseh A, Benkhelifa E, Vouk M, Rindos A. SDIoT: a Software Defined Based Internet of Things Framework. J AMB INTEL HUM COMP. 2015;6(4):453–461.
17. Nguyen TT, Kim DS. Accumulative-Load Aware Routing in Software-Defined Networks. In: 2015 IEEE 13th International Conference on Industrial Informatics . 2015. p. 516–520.
18. Gholami M, Akbari B. Congestion Control in Software Defined Data Center Networks Through Flow Rerouting. In: 2015 23rd Iranian Conference on Electrical Engineering. 2015; p. 654–657.
19. Astaneh SA, Heydari SS. Optimization of SDN Flow Operations in Multi-Failure Restoration Scenarios. IEEE Transactions on Network and Service Management. 2016 Sept;13(3):421–432.
20. Fortz B, Thorup M. Optimizing OSPF/IS-IS weights in a changing world. IEEE journal on selected areas in communications (J-SAC) . 2002 Aug 7;20(4):756-67.
21. Wang SY, Wu CC, Chou CL. Constructing an Optimal Spanning Tree over a Hybrid Network with SDN and Legacy Switches. In: 2015 IEEE Symposium on Computers and Communication (ISCC). 2015; p. 502–507.
22. Cinkler T, Moldovan I, Kern A, Lukovszki C, Sallai G. Optimizing QoS Aware Ethernet Spanning Trees. In: 2005 1st International Conference on Multimedia Services Access Networks. 2005; MSAN '05.; 2005. p. 30–34.
23. Akyildiz IF, Lee A, Wang P, Luo M, Chou W. A Roadmap for Traffic Engineering in SDN-OpenFlow Networks. Computer Networks. 2014;71:1 – 30.
24. Caria M, Jukan A, Hoffmann M. SDN partitioning: A centralized control plane for distributed routing protocols. IEEE Transactions on Network and Service Management(IEEE TNSM) . 2016 Jun 28;13(3):381-93.
25. Nakahodo Y, Naito T, Oki E. Implementation of smart-OSPF in hybrid software-defined network. In2014 4th IEEE International Conference on Network Infrastructure and Digital Content 2014 Sep 19 (pp. 374-378). IEEE.
26. Huang T. Path Computation Enhancement in SDN Networks ;. Master of Applied Science, Program of Computer Networks, Ryerson University, Toronto, Jan. 2015.

Open Access **Baghdad Science Journal** **P-ISSN: 2078-8665**
Published Online First: December 2020 **E-ISSN: 2411-7986**

وكيل توجيه سريان ذكي لتحسين التأخير من النهاية إلى النهاية في الشبكات المعرفة بالبرامجيات

عمر فائز حسين [1] بلال رشيد الكصيم[2] عمر زياد عاكف [3]

[1] قسم ضمان الجودة والأداء الجامعي ، جامعة بغداد، بغداد، العراق.
[2] قسم هندسة الحاسوب ، كلية الهندسة ، الجامعة العراقية، بغداد، العراق.
[3] قسم علوم الحاسوب ، كلية التربية للعلوم الصرفة (إبن الهيثم) ، جامعة بغداد، بغداد، العراق.

الخلاصة :

لضمان الإستجابة للخطأ والإدارة الموزعة، يتم استخدام البروتوكولات الموزعة كأحد المفاهيم المعمارية الرئيسية التي تتضمنها شبكة الإنترنت. ومع ذلك، يمكن التغلب على عدم الكفاءة وعدم الاستقرار والقصور بمساعدة بنية الشبكات الجديدة التي تسمى الشبكات المعرفة بالبرمجيات SDN. الخاصية الرئيسية لهذه المعمارية هي فصل مستوى التحكم عن مستوى البيانات. إن تقليل التصادم سيؤدي إلى تحسين سرعة الإستجابة وزيادة البيانات المرسلة بصورة صحيحة، لهذا السبب يجب أن يكون هناك توزيع متجانس للحمل المروري عبر مسارات الشبكة المختلفة. تقدم هذه الورقة البحثية أداة توجيه ذكية SFSA لتوجيه تدفق البيانات بناءاً على ظروف الشبكة الحالية. لتحسين الإنتاجية وتقليل زمن الوصول، فإن الخوارزمية المقترحة SFSA تقوم بتوزيع حركة مرور البيانات داخل الشبكة على مسارات مناسبة ، بالإضافة إلى الإشراف على الإرتباطات التشعبية وحمل مسارات نقل البيانات. تم استخدام سيناريو خوارزمية توجيه شجرة الامتداد الدنيا MSTوأخرى مع خوارزمية التوجيه المعروفة بفتح أقصر مسار أولاً OSPF لتقييم جودة الخوارمية المقترحة SFSA . على سبيل المقارنة ، بالنسبة لخوارزميات التوجيه المذكروة آنفاً ، فقد حققت استراتيجية SFSAالمقترحة انخفاضاً بنسبة 2٪ في معدل ضياع حزم البيانات PDR ، وبنسبة تتراوح بين 15-45٪ في سرعة إستلام البيانات من المصدر إلى الالوجهة النهائية لحزمة البيانات وكذلك انخفاض بنسبة 23 ٪ في زمن رحلة ذهاب وعودة RTT . تم استخدام محاكي Mininet ووحدة التحكم POX لإجراء المحاكاة. ميزة أخرى من SFSA على MST و OSPF هي أن وقت التنفيذ والاسترداد لا يحمل تقلبات. يتقوم أداة التوجيه الذكية المقترحة في هذه الورقة البحثية من فتح أفقاً جديداً لنشر أدوات ذكية جديدة في شبكة SDN تعزز قابلية برمجة الشبكات وإدارتها .

الكلمات المفتاحية : الشبكات المعرفة بالبرامجيات، الوكيل الذكي، حركة المرور الموجهة، التأخير من النهاية إلى النهاية.