Optimizing Software-Defined Networks with Fuzzy Logic-Based Enhancement of Openflow Protocol

Sajjad H. Hasan[1]*

[1] Department of Computer Techniques Engineering, University of AlKafeel, Al-Najaf, Iraq

Abstract. Today, humans have a strong need to control their devices from a distance so that they can control the world more than before and explore it for various purposes such as how the universe came into being, discovering the way of creation, observing the events in Global situation and so on. Communication with remote devices can be possible in various ways. SDN networks provide a possibility to exchange information between heterogeneous nodes. Considering that in SDN networks, the nodes are very expensive and these nodes themselves are performing many tasks and various vital tasks; Therefore, the cost of each byte of memory occupied on these nodes is very expensive and must be managed in such a way that they have the highest efficiency. Therefore, to solve this problem, it is very necessary and costly to carry out large projects. In the proposed method of this research, by improving the OpenFlow protocol in software-based networks, it is tried to avoid the cooperation of nodes in the directional distribution (not dissemination) of a small data, from the accumulation of extra information in the nodes' memories. Finally, after the simulation, it was observed that the improvement rate of the proposed method has improved by 0.38%, 0.05%, and 0.04%, respectively, compared to RD, FLCFP, and LEACH2013 methods. The improvement rate of the proposed method compared to RD, FLCFP, and LEACH2013 methods was 0.65%, 0.059%, and 0.331%, respectively.

1. Introduction

Software-oriented networks or software-defined network, which is called SDN for short, is a new generation of networks that use virtual layers, virtual switches, central controller, communication standards, and high-level APIs. To perform the control and management tasks of network switches and routers in higher layers by software [1]. SDN systems are an emerging network architecture in which network control is separate from traffic transmission and is programmed directly [2]. This migration to network control, previously limited to network hardware, enables virtual machines and network infrastructure to define and provide new types of services and services, with a new range of applications for greater network flexibility and Broader access to exchanged data, communicate. In other words, SDN reduces the dependence on hardware and increases software capabilities and network intelligence. A software-based controller is responsible for managing how one or more switches send information. The hardware inside the switches only deals with the management of sending traffic according to the rules determined by the controller [2-4].

Software Defined Network (SDN) is proposed as a fundamental change in the network model, by separating the network control from the data level and programmed switching design, and the main and powerful factor of SDN is data flow, widespread deployment in the production and approval of networks. It increases continuously. Also, the openness and ability to program is one of the primary characteristics of data flow. Therefore, establishing security in the real world is very important [5-7].

The OpenFlow protocol was designed by Stanford University in 2008, which was first introduced under the name of Ethan and then under the name of OpenFlow. The OpenFlow protocol was initially used for local networks, which was discussed in the topics related to the future Internet due to the many capabilities it create.

* Corresponding author: sajad.hadi@alkafeel.edu.iq

Also, OpenFlow is a proposed SDN technology to standardize how the controller communicates with network equipment (routers and switches) in an SDN architecture. In fact, OpenFlow provides a description for the migration of the logical form of control from a switch to a central controller. In addition, it also defines a protocol for communication between the controller and the switches [8]. The OpenFlow protocol was designed by Stanford University in 2008, which was first introduced under the name of Ethan and then under the name of OpenFlow. The OpenFlow protocol was initially used for local networks, which was discussed in the topics related to the future Internet due to the many capabilities it created. Also, OpenFlow is a proposed SDN technology to standardize how the controller communicates with network equipment (routers and switches) in an SDN architecture. In fact, OpenFlow provides a description for the migration of the logical form of control from a switch to a central controller. In addition, it also defines a protocol for communication between the controller and the switches [7, 9-11].

Architecture based on Open-Flow has certain capabilities. These features include: software-based traffic analysis, central control, dynamic update of sent rules, and flow abstraction. OpenFlow can simplify traffic management with network flow control and provide global visibility of network flows. However, such precise and adjusted control along with the global monitoring capability of the network comes with a cost [29]. These costs are seen in the phase of establishing flows in the network and also in the phase of statistical collection of network information [12]. Due to these costs, the current design of OpenFlow cannot guarantee the requirements of high-performance networks, because involving OpenFlow to control all traffic flows, especially when the network traffic load is very high, causes overhead. It is added to the central controller and routers and switches in the network. Due to such challenges that exist in the OpenFlow protocol . In this research, an optimal strategy based on fuzzy logic is proposed to improve the performance of OpenFlow protocol and reduce the incurred costs [13].

Undoubtedly, one of the most important problems of SDN networks is the resource limitation of the amount of memory consumed by the nodes. The small size, low weight and the way of placing the contingency that is required by the specific applications of SDN networks make the nodes still depend on techniques for memory management to provide their consumed memory. After all, the efficiency of SDN networks is highly dependent on their lifetime and maintaining their network coverage. Therefore, all the levels of this type of networks should be designed with the knowledge of the amount of memory consumed by the nodes. One of the most important issues regarding these networks is the optimization of the Open Flow protocol. Many efforts have been made to provide methods to optimize the Open Flow protocol in SDN networks. But one of the most widely used and effective methods of optimizing the Open Flow protocol is methods with the lowest delay. The delay-based methods proposed in this research based on fuzzy techniques and rules try to establish memory balance in the network [14].

Several articles have been published in relation to controllers and their management [15-18]. In [19], by reviewing the existing SDN management layers (platform), he identified common management features for SDN networks and further, identified the design requirements of the management layer, which is at the core of the architecture. He also pointed out the open issues and weaknesses of the existing SDN management layers. In [20], an integrated network management and control system framework is proposed. In [21] another research, a distributed hypervisor architecture is presented. which can manage a large number of user flow table control messages. In [22], a complete study has been done in connection with SDN network management and its controller part. In [23], they used an effective method to manage packet buffering in SDN networks and OpenFlow protocol. In their research, they were able to optimize the communication overhead in the devices. In [24], a stochastic model was used to reduce the latency in the OpenFlow protocol of SDN networks.

By reviewing the previous records and the works proposed by other researchers in relation to the topic of this research, it was observed that despite the presentation of various ideas, the methods proposed in the field of improving the OpenFlow protocol in SDN networks still have limitations. In this research, by applying fuzzy logic, the performance of this protocol is improved, including cost reduction, traffic improvement and network congestion, task execution time, and finally response time to sending packets in the network. Therefore, the use of fuzzy logic to improve the OpenFlow protocol is one of the most important innovations of this research project. The desired node is actually a local base node in any situation, which should usually be better than other neighboring nodes in terms of memory and processing power, distance, energy, etc. The desired node receives the data sent by other member nodes and combines them using different methods and sends them to the next node in the form of a smaller packet according to the proposed algorithm. Also, to prevent the rapid emptying of the desired node, it is possible to use the rotation of the role of the desired node between neighboring nodes and distribute the information overhead among other nodes. What is mentioned in most of the methods is that the desired nodes are selected and then other nodes are assigned to it based on the distance with the desired node. Therefore, adjacent nodes will always be in the same position regardless of their memory level.

In this research, the investigation of additional costs and overheads imposed on the network is desired, and the goal is to analyze and design an improved model of OpenFlow version (1.4.0) that can achieve a kind of balance between precise control and statistical collection of information. network and subsequently, a part of the costs will be reduced. Also, in this research, MATLAB tool will be used to provide virtualization and simulation based on OpenFlow.

2. SDN and OpenFlow

Distributed control and transport network protocols running on routers and switches are key technologies that allow information to travel around the world in the form of digital packets. Despite their widespread use, traditional IP networks are complex and difficult to manage. To express desired high-level network policies, network operators must individually configure each network device using low-level, often vendor-specific commands. In addition to configuration complexity, network environments must tolerate fault dynamics and adapt to load changes. Automatic response and reconfiguration mechanisms are virtually non-existent in current IP networks. As a result, it is very challenging to implement the required policies in such a dynamic environment. To add complexity, today's networks are also vertically integrated. The control plane (which decides how to control network traffic) and the data plane (which forwards traffic based on decisions made by the control plane) are co-located in network devices, reducing flexibility and hindering innovation and evolution [25].

The transition from IPv4 to IPv6, which began more than a decade ago and is still largely incomplete, is a testament to this challenge, when in fact IPv6 merely represents a protocol update. Due to the inertia of current IP networks, the complete design, evaluation and deployment of a new routing protocol can take 5-10 years. Likewise, a clean-slate approach to changing the Internet architecture (for example, instead of IP) is considered a daunting task that simply isn't feasible. Finally, this situation has increased the financial and operational costs of implementing an IP network. One of the important consequences of SDN principles is the separation of concerns introduced between defining network policies, implementing them in switch hardware, and forwarding traffic. This separation is the key to desired flexibility, breaking the network control problem into manageable pieces and making it easier to create and introduce new abstractions in networking, simplifying network management and facilitating network evolution and innovation. Some of the advantages that can be achieved in SDN networks are Centralizatio , Optimization , Power Management , Load Balancing , High Availability , Simplicity , Automation , Trouble Shooting, reducing service downtime, applying policy, reducing time , adding new load, productivity, traffic, monitoring, cost reduction, programmable, Abstraction, mobility, quality of service [26].

Although SDN and OpenFlow began as academic experiments, they have gained significant traction in industry over the past few years. Most commercial switch vendors already include OpenFlow API support in their equipment. The SDN movement was strong enough that Google, Facebook, Yahoo, Microsoft established the Open Networking Fund with the goal of promoting and adopting SDN through the development of open standards. As the initial concerns regarding SDN scalability were addressed, specifically the myth that logical centralization implies a physically centralized controller, an issue we address later. SDN ideas have matured and evolved from an academic exercise to a commercial success. Google, for example, has deployed an SDN to connect its data centers around the world. This production network has been used for three years and has helped the company to improve operational efficiency and significantly reduce costs. VMware's network virtualization platform, NSX, is another example. NSX is a commercial solution that provides a fully functional network in software, conditionally independent of the underlying network devices, fully based on SDN principles. As a final example, the world's largest IT companies (from carriers and equipment manufacturers to cloud providers and financial services companies) have recently joined SDN consortia such as ONF and the OpenDaylight initiative, another indication of the importance of SDN from an industry perspective [27].

3. Proposed method

In this section, the full description of the proposed method is presented by providing more details of each part along with the full description of the flowchart of the proposed method and also the proposed algorithm is described. In general, in this chapter, all the flowcharts related to the proposed method are presented and more detailed explanations related to each section are provided. At the end of the chapter, the relevant algorithm is presented in the form of pseudo code. Therefore, the main goal of this research is to introduce, design and analyze an improved method of OpenFlow that can achieve a kind of balance between accurate control and statistical collection of network information in order to reduce the additional overhead on the central controller, routers and switches within the network in major and huge computer networks, to create a tangible reduction in their costs. Figure 1 shows the general architecture of the proposed method. Therefore, in general, according to the proposed architecture, an overview of the operation that is to be performed can be provided.

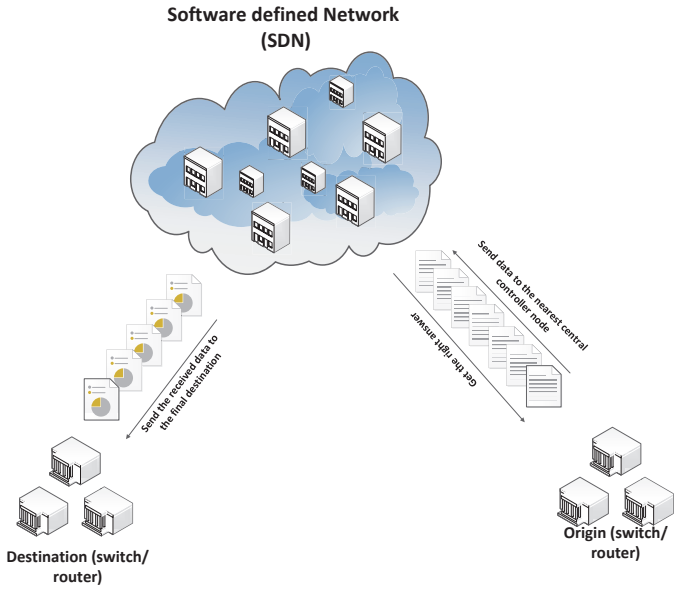

Fig. 1. Architecture of the proposed method

In general, the source node, which is routers or switches or controls in software-oriented networks, is located at one point and sends information with a limited volume in the form of packets with the OpenFlow protocol to the destination node located in a distant or nearby place through other Send router/switch/controls. In the meantime, the strategy used is to use the memory optimally, to prevent the congestion of messages and to ensure the correctness of receiving the message. In this part, according to the goals and nature of the current research, a flowchart has been presented that explains the implementation and performance of the proposed method. Figure 2 shows the general flowchart of the proposed design.

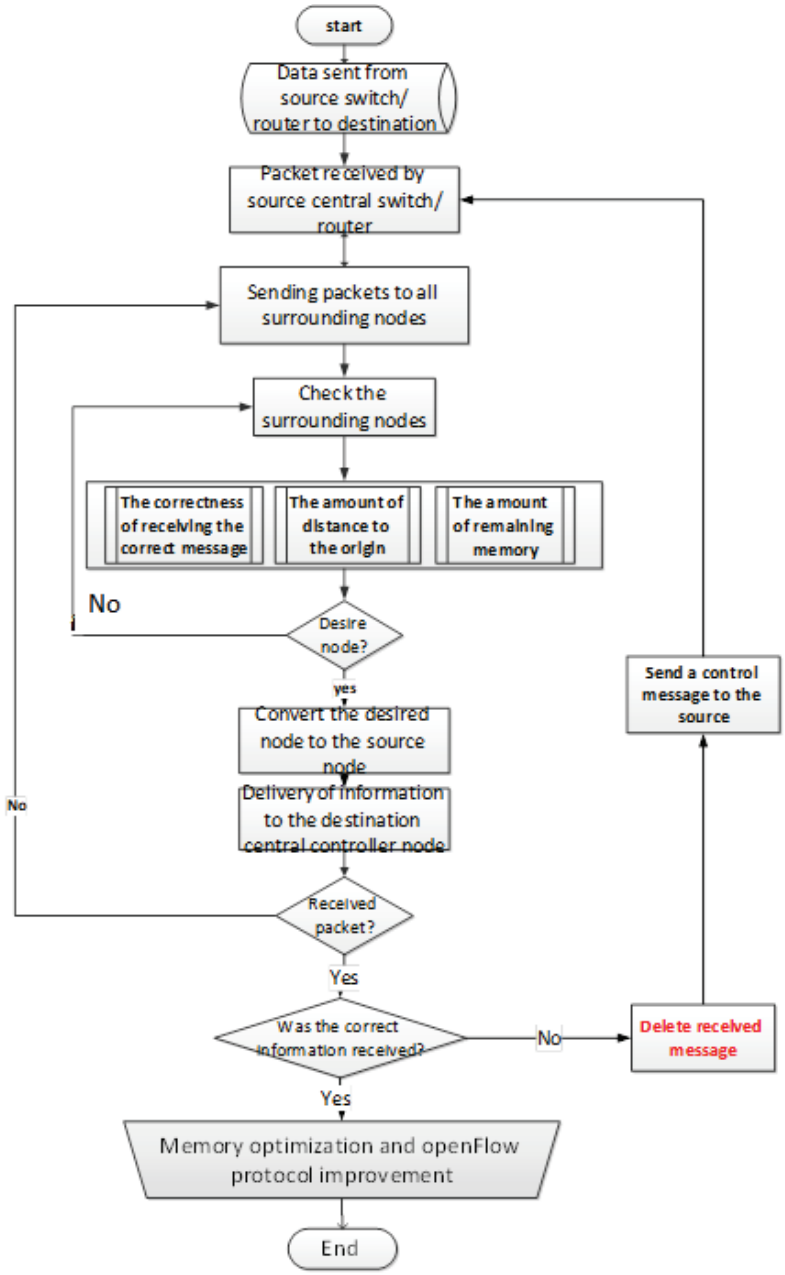

Fig. 2. Flowchart of the proposed method

By examining various articles and the existence of the challenge of information overhead and lack of memory consumption in routers/switches, as well as in accordance with the lack of attention to the updating of information in memories and routing tables of routers and switches in software-based networks in this research, in addition to the importance of this issue, and tried to improve the OpenFlow protocol in order to prevent the delay and congestion of the network. One of the most prominent improvements achieved in the OpenFlow protocol [28] is preventing the accumulation of additional information in the nodes' memories and improving the delay in sending and receiving packets in the network.

Algorithm of the Proposed Method

In this part of the algorithm, which is implemented in the source node and the nodes receiving the message, Figure 3 describes the pseudo code of the proposed method.

```
1. Function My_Approach (N)
2. {
3.  Initialize:
4.  distance=0;
5.  memory=0;
6.  recieve_pck=0;
7.  List<distance>=null;
8.  List<memory>=null;
9.  List<recieve_pck>=null;
10.  Sent_Packet_To_CenterNode (Node src, pocket P);
11.  Neighbors=Get_AllNeighbors(Node src);
12.  For j=1 to N
13.  {
14.   For i=1 to Neighbors.Count
15.    {
16.       List<distance>=get_Distances(Node[j], Neighbors[i]);
17.       List<memory>=get_Memory(Node[j], Neighbors[i]);
18.       List<recieve_pck>=get_Recieve_Pck(Node[j], Neighbors[i]);
19.    }
20.     MainNode=Decision_SYS(List<distance>,List<memory>,List<recieve_pck>);
21.    Send_Control_Message(Node src, Flag msg);
22.     SendMessage_ToNextNode(Message Msg, MainNode);
23.    if (Node[j] ==Des_Node)
24.       {
25.        Break;
26.       }
27.  }
28.  if (Des_Node.RecieveMessage==True)
29.   {
30.          For w=1 to N
31.            {
32.          Update_Memory (Node[w]);
33.          Delete_Msg (Message Msg, Node[w]);
34.       }
35.   }
36.  else
37.   {
38.     Delete_Msg (Message Msg, EndNode);
39.   }
40. }
```

Fig. 3. Algorithm of the proposed method

In line 9-3 some initialization is done. The distance, memory, and accuracy of receiving the message, which is done in the decision-making and checking part of the desired node, are first defined and quantified in this part. The list of distances or memories for each of the neighboring nodes is for each selected node to transmit the message. In line 10, the function Sent_Packet_To_CenterNode is responsible for sending the message from the source to the central node or the next node. In line 11, the corresponding node obtains all the neighboring nodes that are in its range. Neighboring nodes are placed in a list called Neighbors. In line 12, operations are repeated for all nodes in the operating environment. Assuming that the software-based network has N=20, that is, the loop is repeated 20 times per 20 nodes. In line 14, a loop is defined, which is repeated for all neighboring nodes. In lines 16-18, the distance between the neighbor node and the node sending the message, the amount of available memory and the correctness of receiving the message are obtained for each neighboring node. That is, in fact, the source node queries and receives all the above information for all neighboring nodes. Decision logic (line 20) is

used to select an origin node and send data to the destination. The application of fuzzy logic in simulation is that nodes are selected whose memory is optimal and has the closest distance to the source node and also received the message intact. With this strategy, the speed of data transfer is increased and the rate of deleting messages between the way is reduced due to the length of the path, and the power consumption rate of sending messages by routers and routers is also reduced. Therefore, the use of fuzzy logic in sending messages from the source node to the destination node and vice versa seems very necessary and necessary. After the desired node is selected in line 20 using the proposed technique, a control message is sent from the selected node to the previous source node to confirm the correctness of receiving the message (line 21). In line 22, the message is sent to the desired node. Line 23-26 checks whether the node receiving the message is the final destination node or not. If the node is the final destination, the simulation has reached the end and other operations are executed from lines 28 onwards, and otherwise, the previous operation (line 12 onwards) is repeated again until it reaches the final node. Finally, after the corresponding message reaches the end node, it is checked whether the complete message has been received or not. If it is fully received (line 28), all the memory of the nodes in the path that have transmitted the corresponding message are updated (line 32) and the corresponding message is deleted from the memory. Otherwise, the relevant message is also deleted and a non-receipt message is sent to the central source node (line 38).

Simulation steps of the proposed method

In this section, the simulation steps of the proposed method are presented in more detail according to the explanations given in the third chapter. Figure 4 shows the first process of implementing the proposed method, i.e. sending the required data to the main source node.

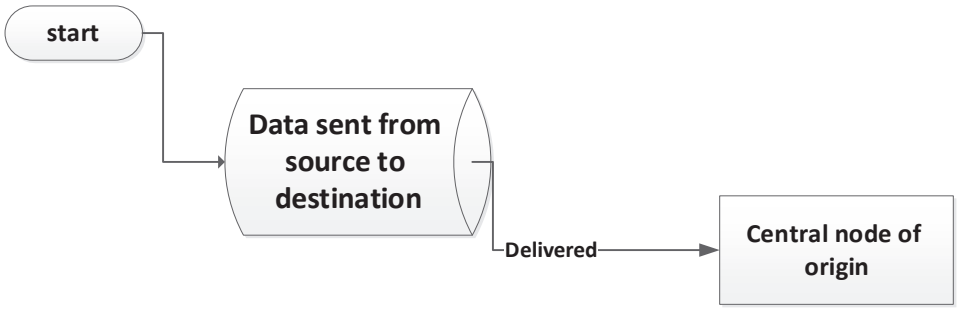

Fig. 4. The first implementation process of the proposed method

As can be seen from the graph in Figure 5, at the beginning of the work, the source, which can be a server, router, hub, and switch, prepares the relevant data that it decides to send to another server, router, hub, and switch. and sends it to the next node in the form of packets. In the simulation done according to Figure 5, the origin and destination nodes are determined.

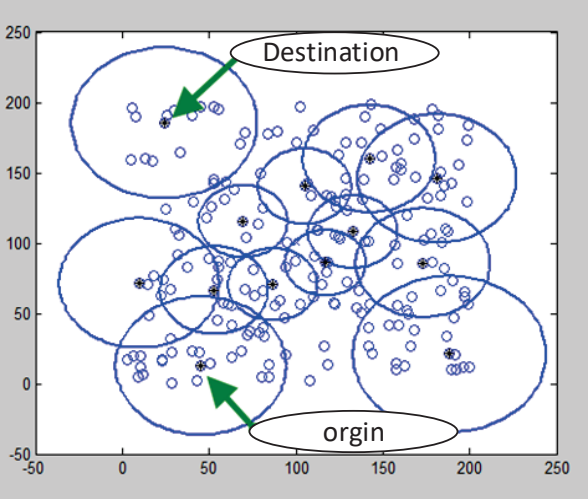

Fig. 5. Determining the origin and destination nodes in the simulation

Figure 6 shows the implementation procedure of the proposed algorithm in the second stage, in which the fuzzy system selects the desired node with the necessary position and ability.

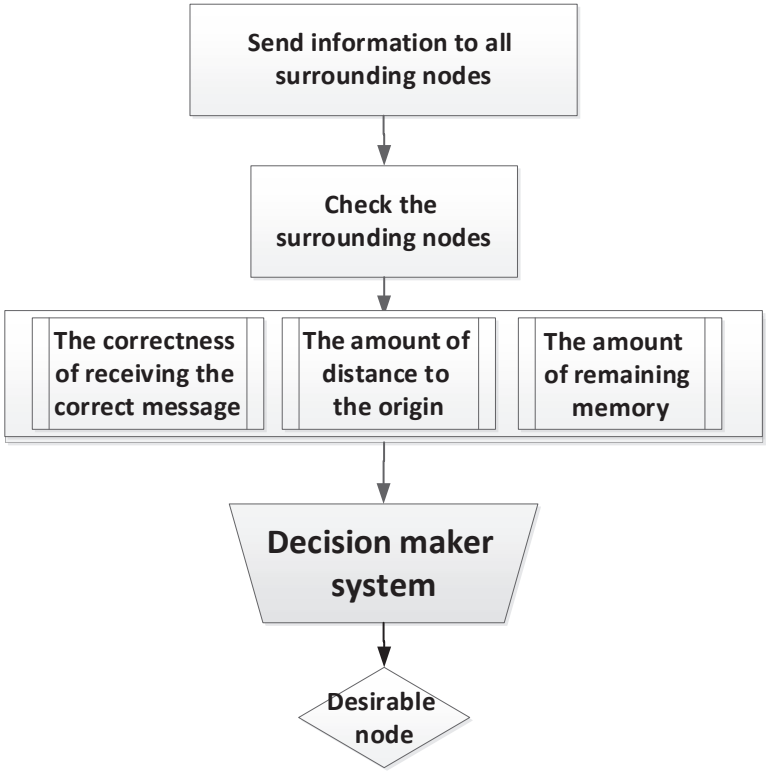

Fig. 6. Fuzzy logic system to select the desired node

As can be seen, the primary node receives the data from the center, which can be an organization or organization, etc., and must deliver the packets to a node that has the necessary conditions to transfer the packet to the next nodes. . In order to receive some of the necessary parameters, the source node has to send the message to all the surrounding nodes so that it can receive information such as distance, amount of free memory, amount of energy and other necessary parameters. Therefore, after sending the control message, it updates the routing table, distance table, etc. so that it can use it in the next steps. In this step, after the information is received by the

surrounding nodes and the corresponding node is sent, the control messages are cleared in the memory to avoid congestion. In the simulation, how to send information to neighboring nodes is shown in the figure below.

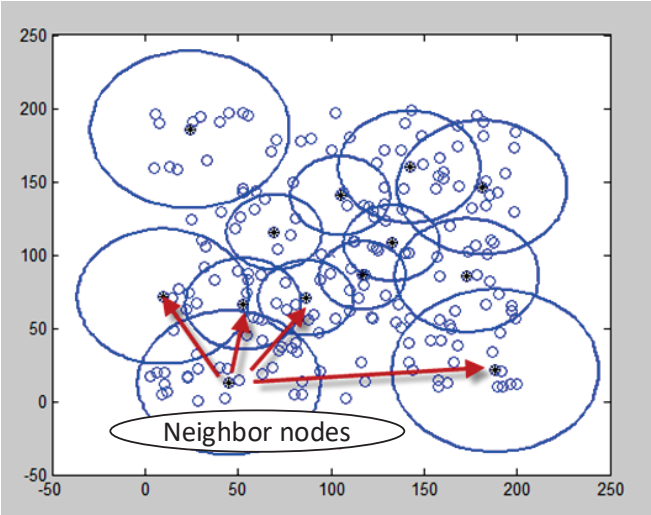

Fig. 7. How to send information to neighboring nodes

Figure 7 shows how to send and receive messages between the source node and other neighboring nodes in Figure 6. As you can see, there is a two-way connection between the source node and other nodes, such as routers, hub switches, etc. Therefore, when the source node receives the information related to the neighboring nodes and updates the desired tables, the decision process is implemented and the desired node is selected based on the parameters of the remaining memory, distance and accuracy percentage of message reception, etc. It should be noted that other parameters can also be considered in fuzzy logic.

Fuzzy system in the proposed method

To select the desired node, we use a series of fuzzy rules as follows. The two main parameters for choosing the desired node by the fuzzy node are the distance of the next node to the current node and the amount of remaining memory. In this section, we put the distance and the amount of remaining memory as the following values:

$$\text{Distance} = \{\text{Low}, \text{Mid}, \text{High}\}$$
$$\text{Amount of remaining memory} = \{\text{Low}, \text{Mid}, \text{High}\}$$

In order to be able to make the right choice, we must form fuzzy rules and make the choice based on these rules.

Table 1. Table of existing rules and conditions for choosing the desired node

Memory distnce	Low	Mid	High
Low	Mid	Good	Best
Mid	Weak	Mid	Good
High	Worst	Weak	Mid

So according to table (1) and the provided fuzzy rules, the method of selection according to distance and battery is defined as Figure 8.

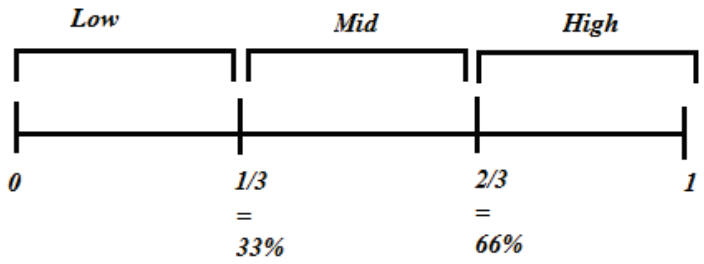

Fig. 8. Polarity factor based on memory

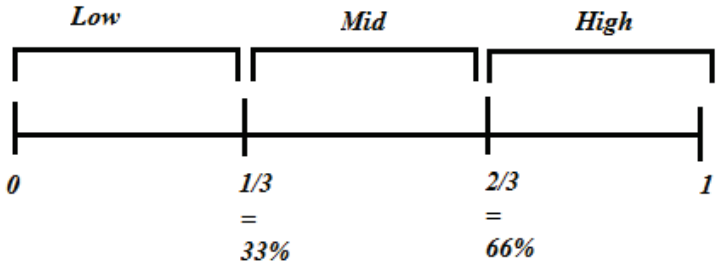

Fig. 9. Polarity factor based on distance

You can choose a suitable node according to the above rules. Assuming that node 1 has 80% remaining memory and node 2 has 60% remaining memory. Therefore, the distance between node 1 and the base station is about 50 meters and the distance between node 2 and the base station is 70 meters. Therefore, with these interpretations, the probability of selecting each node is that because node 1 has an 80% battery and its distance to the base station is 50 meters, as a result, the probability of its selection is high due to its high energy and mid distance, so it is in Good mode. And the probability of choosing node 2 is due to mid energy and High distance in weak mode, and finally node number 1 is selected. The selection of the appropriate node is done by the fuzzy node based on the following conditions:

R1=if (Memory = Low & distance =Low) then select_ node =Mid
R2=if (Memory = Mid & distance =Low) then select_node =Good
R3=if (Memory = High & distance =Low) then select_ node =Best
R4=if (Memory = Low & distance =Mid) then select_ node =Weak
R5=if (Memory = mid & distance =Mid) then select_ node =Mid
R6=if (Memory = High & distance =Mid) then select_ node =Good
R7=if (Memory = Low & distance =High) then select_ node =Worst
R8=if (Memory = Mid & distance = High) then select_ node = weak
R9=if (Memory = High & distance = High) then select_ node = Mid

It is important that if there are two nodes with equal conditions, to select one of them, the fitness function should be used to select the desired node.

$$\text{Fitness} = \text{Memory} + (1 / \text{Distance}) \quad (1)$$
$$\Rightarrow \quad \text{Select-node} = \text{Max}(\text{Fit1}, \text{Fit2})$$

Therefore, by using the proposed method and the presented innovation, a significant improvement in routing, message sending time, and ultimately increasing the node's lifetime will be achieved. It should be noted that the correctness of receiving the message and other necessary criteria are used in the same way in the fuzzy system. According to the simulation, how to choose the desired node is shown in Figure 10.

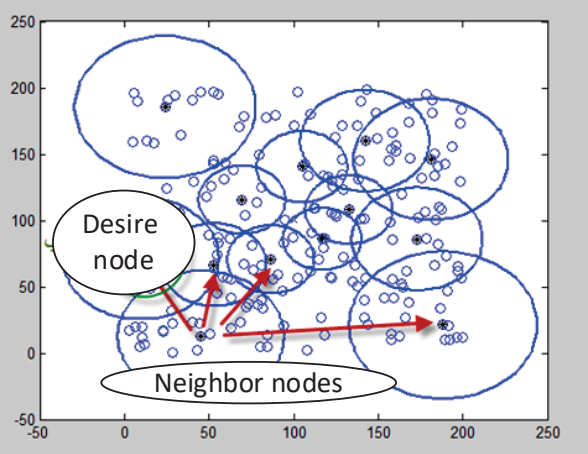

Figure 10. Choosing the desired node in the simulation of the proposed method

Finally, after the desired node is selected, the complete package is sent to the corresponding node and the selected node becomes the source node for transmitting information to other nodes.

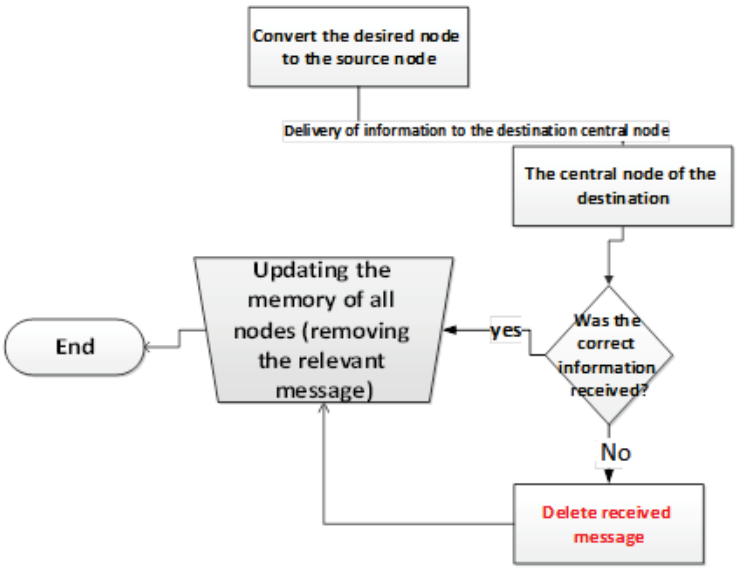

Fig. 11. The final stage of simulation of the proposed method

As can be seen, after the original source node is found, the messages are sent to the destination in the same way. Every time the desired node is sent and selected, it is checked whether the corresponding node is the destination node or not. If the next node is not the destination, the previous process is executed, and if the next node is the destination node, while sending the message to the relevant node, it is checked whether the packet has reached the destination safely or not. If the package does not reach the destination safely, the package is deleted through the node and all nodes are notified to update their memory and finally the corresponding message is deleted from the memory of all nodes. Therefore, if the message reaches the destination safely, the update and notification operation is done, but the message is not deleted from the destination node. In the next section, the results obtained with other methods are compared based on different criteria.

4. Simulation and experimental results

Therefore, according to a system with the above specifications, the relevant simulation has been performed and the results have been evaluated. In this section, the obtained results are explained in full. Table (2) shows the simulation parameters.

Table 2. Simulation parameters and dimensions

Specifications	parameter
1000 nodes	Number of nodes
2000*2000 square meters	Network dimensions
100 KB	The amount of memory nodes
1.40 MHz	The amount of processor
77 KB	The volume of sending packages

As it is clear from Table 2, the simulation parameters are determined so that it can be used in the next steps.

Comparing the results of the proposed method with other methods

According to the simulation done in this part, the proposed method is compared with other methods proposed in recent researches. In general, according to the review of other relevant articles that have worked in this field, three important criteria have been examined. Before describing the evaluation criteria of the proposed method, it should be mentioned that the data related to each of the described methods are in the same conditions as the basic article and have been calculated and compared according to the simulation of the problem.

Message sending delay measure

Figure 12 compares the end-to-end delay for the proposed method and other methods that have been worked on recently. As can be seen, the delay increases in RD (Random Approach), FLCFP(Fuzzy Logic Cluster Formation Protocol) and LCH (Leach 2013)methods; Because some network nodes may send part of the data and have little memory to continue sending and cannot complete the data transfer process or the distance of the selected node is not suitable. In the proposed method, because all the transmissions are done by the node based on the distance criteria, the amount of memory required, the accuracy of receiving the message, etc., the delay in data transmission and transmission is reduced.

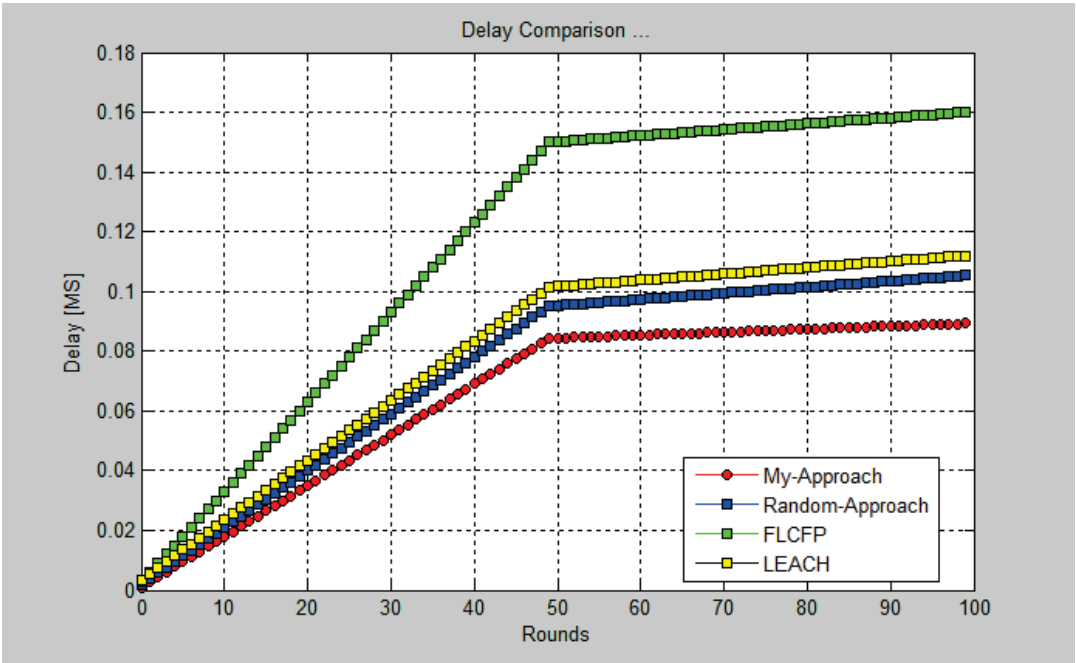

Fig. 12. End-to-end delay of the proposed method and other methods

As it is clear from the figure above, the relevant simulation has been done in different periods. At different times, it can be seen that the amount of packet sending delay in the network has increased significantly in the proposed method compared to other methods to reach a stable level. The increase in delay in the proposed method is very small, and in the following periods, while increasing the number of packets in the network and increasing the traffic, the delay in sending packets in the proposed method is also favorable and does not cause the network to drop.

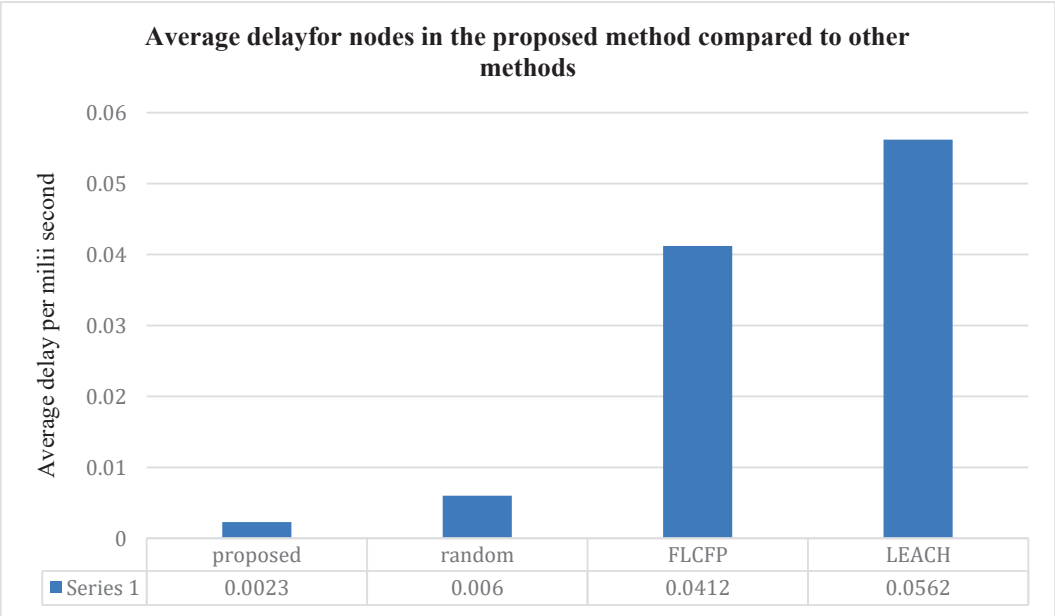

Fig. 13. Average delays in message sending in the proposed method and other methods

As it is clear from Figure 13, the average delay in sending the message of each of the proposed methods, RD method, FLCFP method and LEACH2013 method is shown from bottom to top. Therefore, the improvement rate of the proposed method has improved by 160%, 1700%, and 2369%, respectively, compared to the RD, FLCFP, and LEACH2013 methods.

Amount of memory used

In this research, the memory consumption of the nodes has been investigated according to the application of the fuzzy technique to select the desired nodes. Figure 14 shows the comparison of the amount of memory consumption of nodes according to the number of nodes that are deployed in the SDN network and in a certain period of time with other methods.

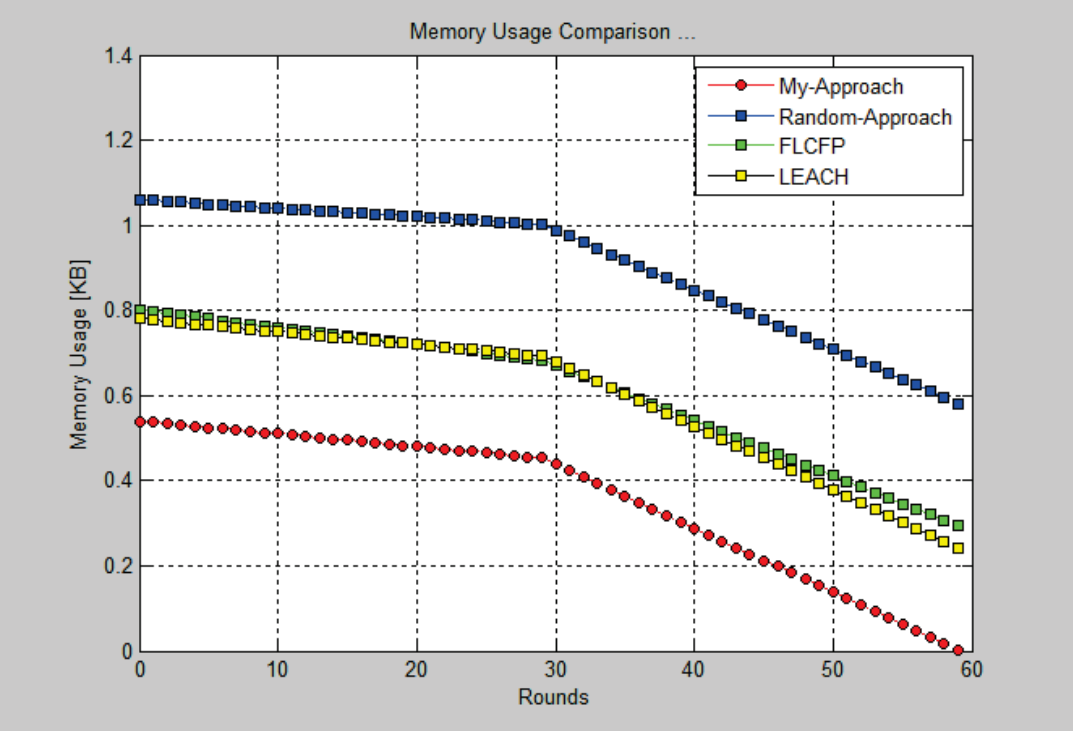

Fig. 14. Comparing the amount of memory consumption of the proposed method compared to other methods

According to the above figure, which has been reviewed and tested in different periods, the amount of memory consumed by the nodes in the proposed method is greatly reduced with the increase of requests and usually reaches a favorable stability. In the proposed method, as the number of packets in the network increases, more memory is freed, and the data that is unused in the memory or has been stored in the memory for a long time is deleted, and finally, the amount of memory consumed by the nodes is reduced. find On the other hand, in the current research, the memory usage mechanism is such that by increasing the number of repeated requests and increasing the number of non-repetitive requests, those processes that are used a lot in the network are kept in the cache and this causes Data overhead can be avoided and much less memory can be saved for more requests. It should be noted that the last number related to the proposed method in the above figure is 0.003 per kilobyte. As it is clear in Figure 14, leach, FLCFP and RD methods are among the methods proposed to reduce memory consumption in SDN networks. Therefore, it can be seen that the proposed method significantly manages the nodes compared to the mentioned methods and consumes little memory, and each node will have the desired message congestion during its lifetime. The figure below shows the average memory consumption for nodes in the proposed method compared to other methods. It should be noted that the amount of memory consumption is based on kilobytes.

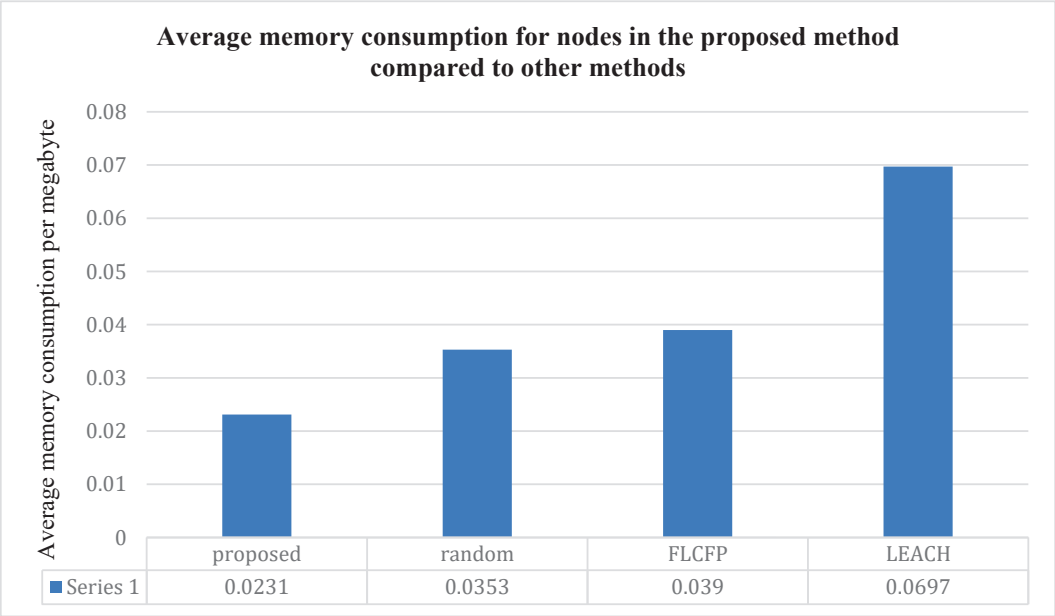

Fig. 15. Average memory consumption for nodes in the proposed method compared to other methods

According to the obtained results, the improvement rate of the proposed method has been improved by 59%, 81% and 218%, respectively, compared to the RD, FLCFP, and LEACH2013 methods, regarding the amount of memory consumption in sending messages and updating.

The number of packets sent per unit of time

Figure 16 compares the sending of information to the neighboring node for the methods of LEACH2013, FLCFP, random broadcast and the proposed method. As can be seen, in the random broadcast mode, LEACH2013, FLCFP, the percentage of sending success is low, because some network nodes may be disabled during the process of sending information, or may not have the necessary conditions to continue transferring data, and finally, the process of transferring information not completed

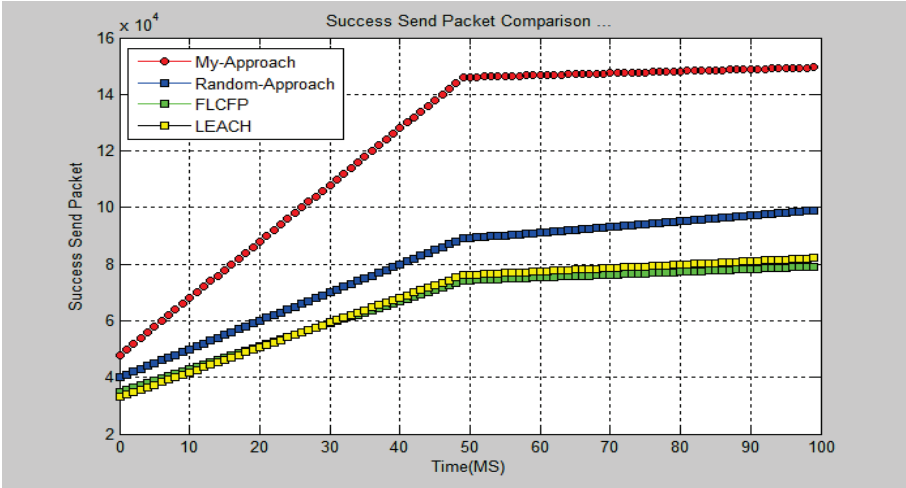

Fig. 16. Comparison of sending rate depending on neighboring nodes per time unit

As can be seen from Figure 16, the proposed method sends a relatively large number of packets in the network at the beginning of the simulation compared to other methods. The amount of sending packets has increased significantly, but this growth trend has reached an optimal level during the simulation and is still better than other methods. The mechanism considered for this part is based on time unit. Figure 17 also shows the average sending rate depending on the neighboring nodes in the proposed method and other methods.

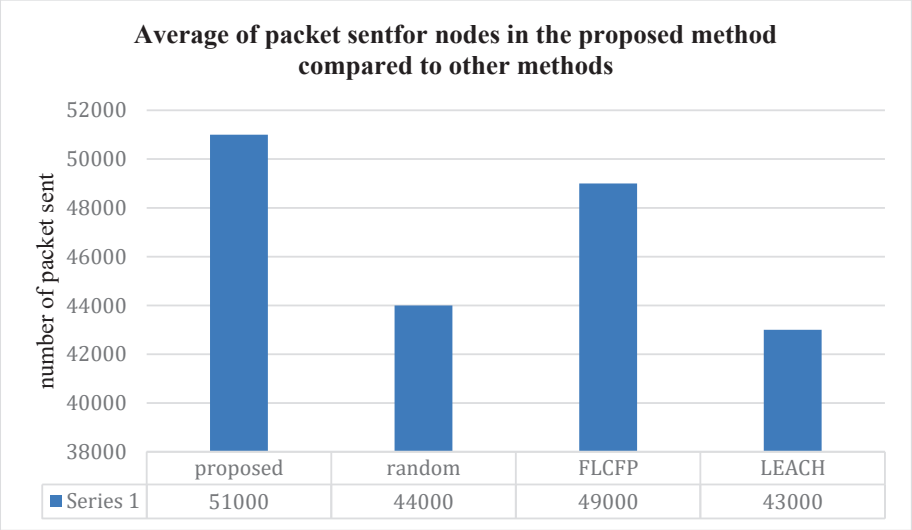

Fig. 17. Average sending rate depending on neighboring nodes per time unit

As it is clear from Figure 17, finally, the improvement rate of the proposed method has been improved by 17%, 41% and 47%, respectively, compared to the RD, FLCFP and LEACH2013 methods according to the average packet sending rate to the neighboring nodes.

5. Conclusion

Despite the progress made in the field of optimization of Open Flow protocol in SDN networks, nodes still rely on optimal methods to provide their memory, optimal routing, reduce congestion and delay due to their large number, small size, and contingent placement method. Also, due to the use of these types of networks in harsh and inaccessible environments, it is not possible to recharge or replace nodes. Therefore, one of the most important issues in SDN networks is the issue of severe memory limitation and optimization of the Open Flow protocol. By examining the different criteria for choosing the desired node and the previous results obtained from it in the proposed method, we came to the conclusion that the combination of these criteria and the use of fuzzy logic is possible, by taking advantage of the advantages of each one and also the possibility of rotating the role of the desired node to distribute Its load will lead to a better effect on the efficiency of the algorithm. The higher the number of desired nodes in the first rounds, the better the routing and optimal transmission of the message to the destination. Finally, after the simulation, it was observed:

- The improvement rate of the proposed method has improved by 0.38%, 0.05%, and 0.04%, respectively, compared to the RD, FLCFP, and LEACH2013 methods.
- The improvement rate of the proposed method has improved by 0.65%, 0.059%, and 0.331%, respectively, compared to the RD, FLCFP, and LEACH2013 methods, according to the amount of memory consumption in sending messages.
- The improvement rate of the proposed method has been improved according to the rate of successful message sending compared to RD, FLCFP and LEACH2013 methods, respectively.

Refrences

1. K. Greene, "TR10: Software-defined networking," ed: MIT Technology Review Cambridge, MA, USA, 2009.
2. M. Harouni, M. Karimi, A. Nasr, H. Mahmoudi, and Z. Arab Najafabadi, "Health monitoring methods in heart diseases based on data mining approach: A directional review," in Prognostic models in healthcare: Ai and statistical approaches: Springer, 2022, pp. 115-159.
3. B. Lakshmi Narayan, S. Rai, and P. N. Hamsavath, "Network De-materialization in Open Flow Network."
4. M. Bhuyan, S. Kashihara, D. Fall, Y. Taenaka, and Y. Kadobayashi, "A survey on blockchain, SDN and NFV for the smart-home security," Internet of Things, p. 100588, 2022.
5. M. H. Rehmani, A. Davy, B. Jennings, and C. Assi, "Software defined networks-based smart grid communication: A comprehensive survey," IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2637-2670, 2019.
6. A. Raftarai, R. R. Mahounaki, M. Harouni, M. Karimi, and S. K. Olghoran, "Predictive models of hospital readmission rate using the improved AdaBoost in COVID-19," in Intelligent Computing Applications for COVID-19: CRC Press, 2021, pp. 67-86.
7. A. J. Moshayedi et al., "E-Nose design and structures from statistical analysis to application in robotic: a compressive review," EAI Endorsed Transactions on AI and Robotics, vol. 2, no. 1, pp. e1-e1, 2023.
8. A. Moubayed, A. Refaey, and A. Shami, "Software-defined perimeter (sdp): State of the art secure solution for modern networks," IEEE network, vol. 33, no. 5, pp. 226-233, 2019.
9. Z. Latif, K. Sharif, F. Li, M. M. Karim, S. Biswas, and Y. Wang, "A comprehensive survey of interface protocols for software defined networks," Journal of Network and Computer Applications, vol. 156, p. 102563, 2020.
10. B. Goswami, M. Kulkarni, and J. Paulose, "A Survey on P4 Challenges in Software Defined Networks: P4 Programming," IEEE Access, 2023.
11. S. Azodolmolky, Software defined networking with OpenFlow. Packt Publishing, 2013.
12. R. Swami, M. Dave, and V. Ranga, "Software-defined networking-based DDoS defense mechanisms," ACM Computing Surveys (CSUR), vol. 52, no. 2, pp. 1-36, 2019.
13. M. Chouikik, M. Ouaissa, M. Ouaissa, Z. Boulouard, and M. Kissi, "Impact of DoS attacks in software defined networks," in AIP Conference Proceedings, 2023, vol. 2814, no. 1: AIP Publishing.
14. J. López, C. Chatzinakis, M. Cartigny, and C. Poletti, "Software defined networking flow admission and routing under minimal security constraints," arXiv preprint arXiv:2307.11879, 2023.
15. J. Goor, "Log Parsing in Software-Defined Networking to generate DyNetKAT models," University of Twente, 2023.
16. A. Nunez, J. Ayoka, M. Z. Islam, and P. Ruiz, "A Brief Overview of Software-Defined Networking," arXiv preprint arXiv:2302.00165, 2023.
17. S. K. Keshari, V. Kansal, S. Kumar, and N. R. Roy, "A Review of Deterministic and Non-deterministic Load Balancing Mechanisms in Software Defined Networks," in 2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2023, pp. 305-310: IEEE.
18. M. Hassan, M. Gregory, and S. Li, "Multi-Domain Federation utilising Software Defined Networking: a Review," IEEE Access, 2023.
19. Y. Wang and I. Matta, "Sdn management layer: Design requirements and future direction," in 2014 IEEE 22nd International Conference on Network Protocols, 2014, pp. 555-562: IEEE.
20. J. Ali, R. H. Jhaveri, M. Alswailim, and B.-h. Roh, "ESCALB: An effective slave controller allocation-based load balancing scheme for multi-domain SDN-enabled-IoT networks," Journal of King Saud University-Computer and Information Sciences, vol. 35, no. 6, p. 101566, 2023.
21. J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic Network Virtualization and Pervasive Network Intelligence for 6G," arXiv preprint arXiv:2301.00519, 2023.
22. L. Golightly, P. Modesti, R. Garcia, and V. Chang, "Securing Distributed Systems: A Survey on Access Control Techniques for Cloud, Blockchain, IoT and SDN," Cyber Security and Applications, p. 100015, 2023.
23. J. Mao, B. Han, Z. Sun, X. Lu, and Z. Zhang, "Efficient mismatched packet buffer management with packet order-preserving for OpenFlow networks," Computer Networks, vol. 110, pp. 91-103, 2016.
24. U. Javed, A. Iqbal, S. Saleh, S. A. Haider, and M. U. Ilyas, "A stochastic model for transit latency in OpenFlow SDNs," Computer Networks, vol. 113, pp. 218-229, 2017.
25. A. Liatifis, P. Sarigiannidis, V. Argyriou, and T. Lagkas, "Advancing sdn from openflow to p4: A survey," ACM Computing Surveys, vol. 55, no. 9, pp. 1-37, 2023.
26. N. Satheesh et al., "Flow-based anomaly intrusion detection using machine learning model with software defined networking for OpenFlow network," Microprocessors and Microsystems, vol. 79, p. 103285, 2020.

27. R. Wazirali, R. Ahmad, and S. Alhiyari, "SDN-openflow topology discovery: An overview of performance issues," Applied Sciences, vol. 11, no. 15, p. 6999, 2021.
28. J. R. de Almeida Amazonas, G. Santos-Boada, and J. Solé-Pareta, "A critical review of OpenFlow/SDN-based networks," in 2014 16th International Conference on Transparent Optical Networks (ICTON), 2014, pp. 1-5: IEEE.