An Autonomic Software Defined Network (SDN) Architecture With Performance Improvement Considering

Alireza Shirmarz*
Department of Computer Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran.
A.shirmarz@iau-tnb.ac.ir
Ali Ghaffari
Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran
A.Ghaffari@iaut.ac.ir

Received: 04/Feb/2020 Revised: 09/Mar/2020 Accepted: 16/Apr/2020

Abstract
SDN makes the network programmable, agile, and flexible with data and control traffic separating. This architecture consists of three layers which are application, control and data. The aim of our research is concentrated on the control layer to improve the performance of the network in an autonomic manner. In the first step, we have categorized the performance improvement researches based on network performance improvement solutions proposed in the recent papers. This performance improvement solution clustering is one of our contributions to our paper. The significant contribution in this paper is a novel autonomic SDN-based architecture to ameliorate the performance metrics including blocking probability (BP), delay, jitter, packet loss rate (PLR), and path utilization. Our SDN-based autonomic system consists of three layers (data, autonomic control, and Route learning) to separate the traffics based on deep neural networks (DNN) and to route the flows with the greedy algorithm. The autonomic SDN-based architecture which has proposed in this paper makes better network performance metrics dynamically. Our proposed autonomic architecture will be developed in the POX controller which has developed by python. Mininet is used for simulation and the results are compared with the commonly used SDN named pure SDN in this article. The simulation results show that our structure works better in a full-mesh topology and improves the performance metrics simultaneously. The average performance is improved by about %2.5 in comparison with pure SDN architecture based on the Area Under Curve (AUC) of network performance.

Keywords: SDN; Performance; Autonomic System; Resource Management; QoS.

1- Introduction

The number of internet users is growing rapidly, so the network resources will be restricted for Internet service providing in the future. According to this Internet growth, different architectures have been proposed in various scrutinies [1]. The resource shortage can give rise to the service provisioning problem in the networks and causes network equipment configuration complexity as mentioned in [2]. Based on the future network requirements, SDN has been proposed as a programmable architecture [3]. SDN is an architecture that has been investigated in many papers [4][5][6][7]. SDN architecture consists of three layers (APP, Control, and Data) and three APIs (Northbound, Southbound, and East-West) which are shown in Fig.1, briefly.
Data plane is composed of FEs which is simple forwarding elements, control plane ha the role of decision making in SDN and all FEs send their flows' first packet of flows in the packet-in message to the controllers, the controllers impose the flow entries with suitable action to the Fes flow tables. The controller can be single centralized or conceptual centeralized. The Conceptual centralized controller is composed of some controller which are related together with east-west APIs. The data plane and control plane are connected by southbound API. The application layer is based on network application and is connected with the controller with northbound APIs.

* Corresponding Author

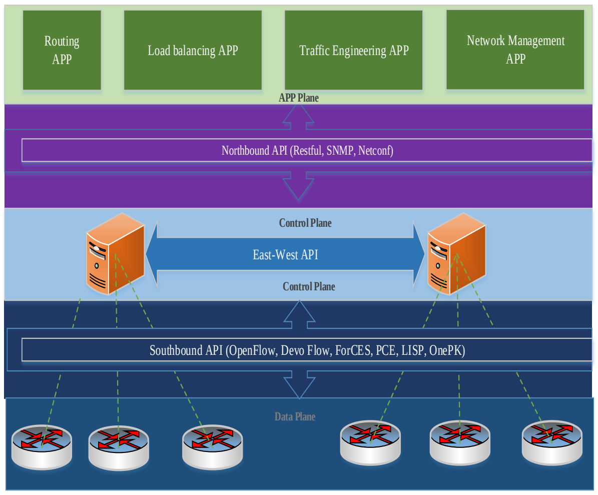

Fig.1 SDN Architecture [8]

We have surveyed the performance improvement in SDN and published our research in [8]. According to our published survey, the solutions for SDN performance improvement have been categorized into three groups that are DC & Cloud, Wireless, and WAN. The SDN-based network performance metrics which have been used are delay, throughput, delay, jitter, packet loss rate (PLR), blocking probability (BP), flow rate (FR), bit error rate (BER), and energy. The main question in this paper is how to improve software defined network’s performance in an autonomic manner? Which we solve this problem with network performance improvement approaches categorization in SDN and a novel autonomic SDN-based architecture to cover our problem and improve the network performance metrics simultaneously.
In this paper, we investigate the solutions and propose an SDN-based architecture to make dedicated performance metrics improved. Finally, this our proposed architecture's advantages and drawbacks will be discussed in comparison with other ones

2- Related Works

The performance improvement in SDN is a problem which has been paid attention in [8]. The paper has extracted performance metrics and solutions in three dedicated SDN-based networks. This paper is the continuing work of that survey which was published. The other solutions for Quality of Service (QoS) in SDN have been investigated in [9]. The authors have proposed a new SDN-based architecture to provide QoS for application with the use of SDN capabilities. Omar Aldhaibani et al have suggested an SDN-based architecture for the Handover (HO) decision in Wi-Fi environments in [10]. They have used SDN to make hand-over smart in IEEE 802.11 wireless LAN and proved that the QoE (Quality of Experience) has been improved. Feng et al have analyzed the applications of SDN to different types of wireless networks. They have discussed the performance improvement in SDN-based wireless networks and presented future direction in SDWN (Software Defined Wireless Network) [11]. T. Shozi et al have worked on an SDN-based overlay solution for the existing traditional Wide Area Networks (WAN) environment, targeting provisioning flexibility and control in case of network failures, through a distributed SDN network overlay and edge SDN devices [12]. The goal was to facilitate the transition to SDN in developing economies through the adoption of SDN while retaining legacy infrastructure. We present an SDN overlay on top of the existing WAN using the South African National Research Network (SANReN) network as a use case. Kleinrouweler et al have worked on Dynamic adaptive streaming over HTTP (DASH) which is a simple, but effective, technology for video streaming over the Internet in [13]. They proposed an SDN-based architecture for DASH-aware networking that also enables internet service providers, network administrators, and end-users to configure their networks to their requirements. In paper [14], Autonomic QoS Management Mechanism in Software Defined Network has been proposed to improve QoS with open flow protocol modification. G. Poulios et al have proposed an SDN-based architecture for Self-Organization networking (SON) in LTE [15]. Pedro Neves et al have worked on 5G and suggested an architecture with SDN capabilities in [16]. These papers have proposed SDN-based architecture in a dedicated use case, so we analyze the performance improvement solutions in SDN to propose an innovative architecture to improve network performance with the SDN concept.

3- SDN Performance Improvement Solutions Analysis

In this paper, in continuing our survey in [8], we analyze the researches to extract performance improvement in SDN and according to the recent researches, we can divide the solutions in a tree which is shown in Fig 2 in summary. The main resource which is used to control the network performance is bandwidth. These solutions are divided into two significant taxonomies which are Link Capacity Expansion, and bandwidth allocation management which are the goals of different researches to ameliorate network performance metrics.

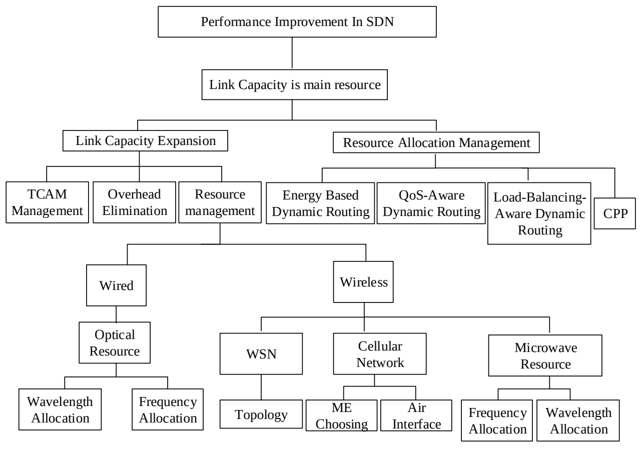

Fig. 2 Performance improvement solutions taxonomy

3-1- Link Capacity Expansion

The expansion of link capacity is the aim of some researches that seek to develop bandwidth capacity with TCAM management, overhead elimination, and link resource management.

- TCAM management: This solution concentrates on memory space management which causes performance improvement. Switches, routers, and Forwarding Elements (FEs) use the ternary content-addressable memory (TCAM) which is fast in reading/writing speed, but this memory type is expensive.
- Overhead Elimination: This solution is used to eliminate the overhead of reading/writing protocols in southbound API. The most southbound protocols which have been used in the researches are open-source which is open flow.
- Resource Management: The links in the network can be wired or wireless. The resources which have been used in wired media are optical one and the researchers increase bandwidth with frequency and wavelength management. In wireless media, the resources are different based on network applications. In WSN, topology is managed, in the cellular network, air interface and ME selection are managed, and in microwave-based resources like frequency and wavelength are used to control.

3-2- Bandwidth Allocation Management

In this solution, the researchers work to manage resources with awareness. The awareness types discriminate the solutions.

- Energy-based Dynamic Routing: this solution tries to route the traffic with energy consumption consideration, so the energy will be improved.
- QoS-aware Dynamic Routing: this solution routes the flows based on QoS metrics consideration. This method betters QoS.
- Load-balancing Aware Dynamic Routing: this method routes the flows based on the load in FEs and Servers. This one decreases congestion and improve performance measures.
- Controller Placement Problem (CPP): this way works on improvement in performance measures with controlling of the number of controllers and controllers place.

This section analyzes the solutions which have been proposed for performance improvement in SDN to design the architecture for performance improvement in SDN. In the following, our proposed architecture will be suggested.

4- Proposed Architecture for Autonomic Performance Improvement in SDN

This architecture is inspired by the IBM architecture which is presented in [17]. This architecture makes the network autonomic and consists of four main elements as shown in Fig 3. This model is used as a reference model in our proposed architecture.

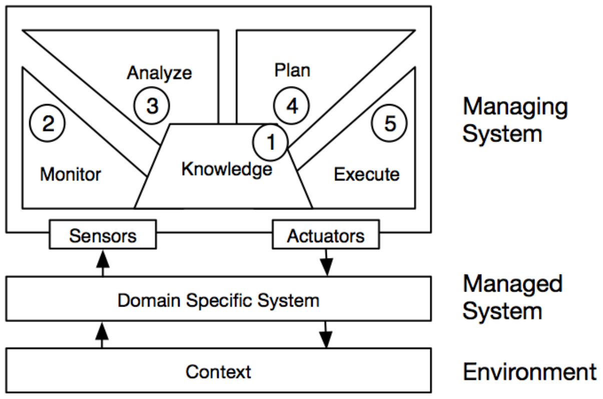

Fig. 3 Reference Model for Autonomic System [17]

- Knowledge: this is a database that is collected from the system and makes the required knowledge to make the autonomic system.
- Monitor: this is composed of the monitoring process
- Analyze: this section is the module for analyzing the system to make a plan for the system
- Plan: this module makes the plan for execution

- Execute: this module imposes the plan to the system

SDN makes the network programmable and agile. It also causes the network flexibility. We propose an architecture that is composed of SDN and MAPE-K for performance improvement. Our proposed architecture should be an autonomic one to support the performance metrics in ISPs and network service providers.

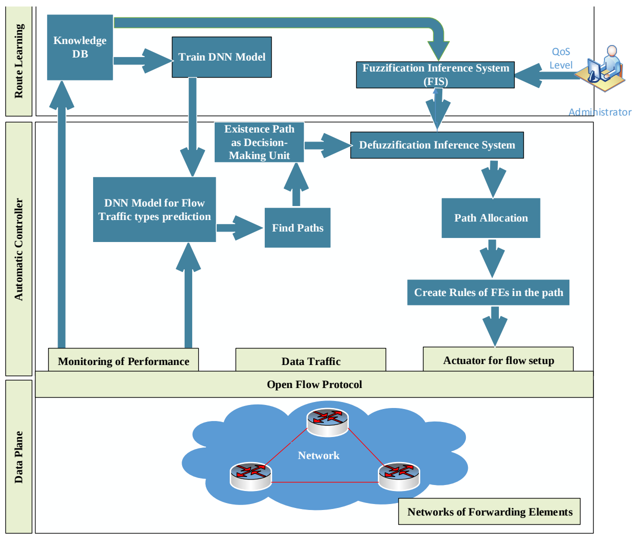
Route Learning
Knowledge DB
Train DNN Model
Fuzzification Inference System (FIS)
QoS Level
Administrator
Automatic Controller
Existence Path as Decision-Making Unit
Defuzzification Inference System
DNN Model for Flow Traffic types prediction
Find Paths
Path Allocation
Create Rules of FEs in the path
Monitoring of Performance
Data Traffic
Actuator for flow setup
Open Flow Protocol
Data Plane
Network
Networks of Forwarding Elements

Fig. 4 Proposed autonomic architecture for performance improvement in SDN

This autonomic system provides the performance requirements automatically and involuntarily. The performance metrics include blocking probability (BP), delay, jitter, packet loss rate (PLR), and Path utilization which are monitored and analyzed in our scheme.

- Blocking Probability (BP): this metric shows the percentage of blocked flow requests in the controller.
- Delay: this metric indicates the delay of the path which contains propagation, processing, queuing, and transmission delay
- Jitter: it is the variance of delay
- Packet loss rate (PLR): this is the percentage of packets lost in one second.
- Path Utilization: this metric shows the percentage of path capacity which has been used in the network.

QoS separates the traffics in the link layer, network layer, and transport layer to make sure that the expected quality of applications will be provided by the network with a variety of required resources. On the contrary, network performance should be set out independently and without awareness of applications and services, to a dependable network.

Our proposed architecture has three layers which are data, Autonomic control, and route learning layers.

- Data layer: this layer indicates the data which is passing through the network
- Autonomic control layer: this layer contains the controller with some modules to make the controller as an autonomic one.
- Route learning layer: this layer gathers the information and trains the model periodically till makes the machine-learning flow discriminator more accurate.

In the next sub-section, the modules of each layer will be expressed.

4-1- Data Layer

In this layer, packets send the first packets of the flows and the non-dedicated rule packets are sent to the controller as a packet-in message format of open flow ver1.3 as mentioned in Fig 5.

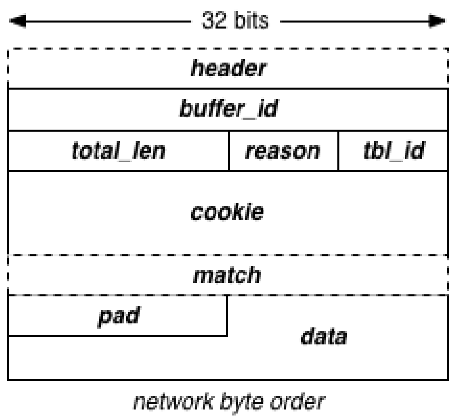

Fig. 5 Packet-in format in Open flow ver1.3 [18]

The flow is a sequence of packets that have the same source and destination. The unique source and destination are specified by the unified source and destination MAC, IP, and destination port number. The controller, based on the flow information routes the flow path that will be discussed in the controller subsection. The other information which is needed to collect will be sent to the route learning layer for knowledge database collection to increase the training samples.

4-2- Autonomic Control Layer

This layer has 4 main modules and one common module which has discussed here.

- DNN Model for traffic types' discrimination: this module used a deep neural network (DNN) to classify the traffic types automatically. This module makes our architecture independent from the applications. This model needs the training set which had been gathered in the route learning layer and trained there before. This Module is implemented using the paper [19].

- Find Path Module: a module is needed to find the path with source and destination information that had been extracted from the packet-in message. This module constructs the existing paths in which there are between source and destination.
- Path Allocation Module: This module allocates the proper path to the flow requests with the performance considering. All existing paths are sorted based on the performance metrics and select the path with the highest performance.
- Create Rules for the FEs: the selection path needs the configuration of all FEs which exist in the path, so this module makes ready all flow entity for specified FE. Finally, all configurations will be imposed on all FEs.

The autonomic controller has been implemented in our work in [20]. In the next subsection, the Route learning layer will be discussed. The tasks which have been don is shown in Algorithm 1.

Algorithm.1: Steps of routing based on the greedy algorithm in [20]

```
Algorithm 1: The greedy adaptive flow routing algorithm

1.  Input→ F as the requested flows set
2.  Sort F based on $B^f$ descending
3.  For each flow in F
      Begin for F
4.  Find all existing paths between current flow source and
      destination with Findpaths function
5.  Sort the paths of the current flow ascending
6.  For each path in paths
        Begin for paths
7.      If the current path supports the $B^f_{src,dst}$ and
        $D^f_{src,dst}$ and $J^f_{src,dst}$ and $L^f_{src,dst}$ as the current flow
        performance thresholds
          Begin if
            Allocate the current flow to the current path and
            return the current path
            Goto the next flow if exists
          End if
          Else
            Goto another path if exists, otherwise return the
            null value for the current flow
          End Else

        End for paths
      End for F
8.  Output→ a path or null for each flow in F
```

4-3- Autonomic Control Layer

This layer has two significant roles in our proposed architecture, at first, it is used to information collection and model training, and the second is a channel for communication with network admins.

- Training Role: the required information is gathered from the data layer and stored in a PostgreSQL database and the data is analyzed and store the number of packets in each flow, the number of bytes in each flow, the number of blocked flow, and so on. The suitable action is tagged and trained with Moore traffics which is a reference dataset in ten days [21].
- Network Admins Interface: this module communicates with the human being, so we use fuzzy logic to show the network performance situation in 5 degrees which are very bad, bad, average, good, and very good. These states show the state of the performance of the networks linguistically. The other option in the admin interface is the prioritization between BP and QoE (Quality of Experience) that should be defined by the network admins. This option can prioritize that if there is bandwidth capacity resource more than required and the network situation is good or very good the network can improve the quality of experience or decrease the BP. It depends on the policy of networks.

This algorithm is shown in Algorithm 2.

```
Algorithm 2 Training algorithm [19]

1.  Input→$X_L$: The labelled training set
          $X_U$: The unlabeled training set
2.  Randomly initialize the stacked autoencoder
3.  For $X_{U1} \in X_U$, do
        Calculate the output of the first hidden layer y($X_{U1}$)
        Calculate z($X_{U1}$) by formula
    End for
4.  Compute the weight and bias of the first hidden layer by
    minimizing $LS(\theta_1)$
5.  For i=2 to M, do
        Use the output of the (i-1)th hidden layer as the input
        of the ith
        Hidden layer to train the ith hidden layer by minimizing $LS(\theta_1)$
    End for
6.  Randomly initialize the softmax regression layer
7.  Train softmax regression layer to achieve the minimization of
    $LS(\theta_2)$ by using the output of the final hidden layer as the input
8.  Initialize the hybrid deep learning network with obtained
    Parameters through the pre-training process
9.  Fine-tune the hybrid deep network to minimize $LSS(\theta)$ by BP
    algorithm
10. Output→ $\theta$: The weights and bias of all layers
```

This architecture is developed in the POX controller to show its behaviour in comparison with the pure SDN one. Three scenarios are defined to show that our architecture has better performance in comparison with pure SDN.

The architecture of a deep learning-based application classifier is shown in Fig 6.

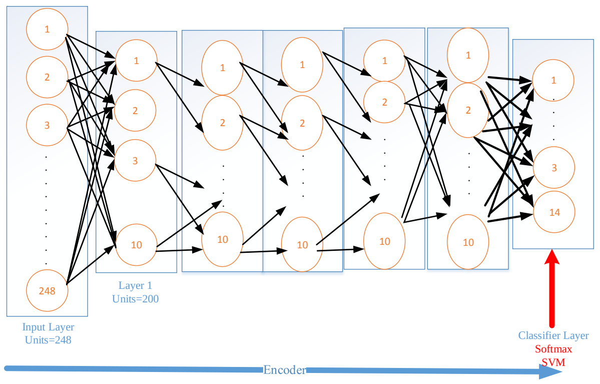

Fig. 6 Application Classifier based on Deep Learning Architecture

5- Experiments and Results

Our proposed architecture is implemented in the POX controller and Mininet. The requirements for simulation are presented in Table 1.

Table 1: Simulation Requirements

Hardware/ Software	Framework	specs
Hardware	CPU	Intel(R) Core(TM)-i7-2.4GH
	Ram	12 GB
	Hard	2 TB
Software	OS	Windows 10-64bit
	Programming language	Python 2.7
	Programming IDE	Spyder 3.3.3
	Deep learning software	Tensorflow/python
	Deep learning library	Keras
	Machine Learning library	scikit-learn v0.21.3
	Controller	Pox 0.2.0

The simulation runs for 15 min with 10 times iteration. The average BP, delay, jitter, PLR, and utilization will be considered and normalize with the Eq.(1).

$$Normalize(x) = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

Three scenarios are defined with a different number of nodes and links which will be defined in the following subsection.

5-1- Simulation Scenarios

Three scenarios are defined to show the application of our proposed architecture with performance improvement considering.

- Scenario-1: This scenario is full-mesh one which has specs and the limitation value which is selected randomly as mentioned in Table 2.

Table 2: Full-mesh scenario

Specs	Limitation of value
The number of nodes	9
The number of links	36
The number of flow requests	40
Link speed	10-100 Mbps
Link delay	10-120 ms
Link jitter	10-20ms
Link PLR	1%-7%

- Scenario-2: This scenario is partial-mesh one which has specs and the limitation value which is selected randomly as mentioned in Table 3.

Table 3: Full-mesh scenario

Specs	Limitation of value
The number of nodes	9
The number of links	24
The number of flow requests	40
Link speed	10-100 Mbps
Link delay	10-120 ms
Link jitter	10-20ms
Link PLR	1%-7%

- Scenario-3: This scenario is sparse one which has specs and the limitation value which is selected randomly as mentioned in Table 4.

Table 4: Full-mesh scenario

Specs	Limitation of value
The number of nodes	9
The number of links	9
The number of flow requests	40
Link speed	10-100 Mbps
Link delay	10-120 ms
Link jitter	10-20ms
Link PLR	1%-7%

All values are assigned randomly in each scenario between min and max value which have been determined in Tables 2, 3, and 4.

5-2- Evaluation

For evaluation, we do the algorithm which is shown in Algorithm 2.

```
Algorithm 2: The evaluation of two architectures
1. Input→Pure SDN | Proposed Architecture
2. Run simulation for 15 minutes
3. Sampling every 5 seconds the performance metrics
4. average of BP, delay, jitter, PLR, utilization for 15 minutes
5. Normalize the performance metrics
6. complementary of utilization is calculated
7.  Draw the AUC for 5 performance metrics
8. Calculate the area of the curve for each scenario
9. Output→ comparison of Proposed architecture and pure SDN
```

To evaluate the proposed autonomic SDN-based architecture which has been developed in POX as the controller and Mininet as an emulator the AUC is used. AUC considers all performance metrics that should be minimized and utilization which should be maximized. We use complementary utilization that can minimize. The AUC is used to show performance improvement in our proposed autonomic SDN-based architecture in collation with pure SDN with OpenFlow spanning tree in POX to prevent a loop.

The average results of simulation in five performance metrics are shown in Fig 7, Fig 8, and Fig 9 as the AUC chart for scenario-1, scenario-2, and scenario-3 respectively. All metrics should be minimized, but the utilization should maximize, so the complementary of utilization is considered as 1-U.

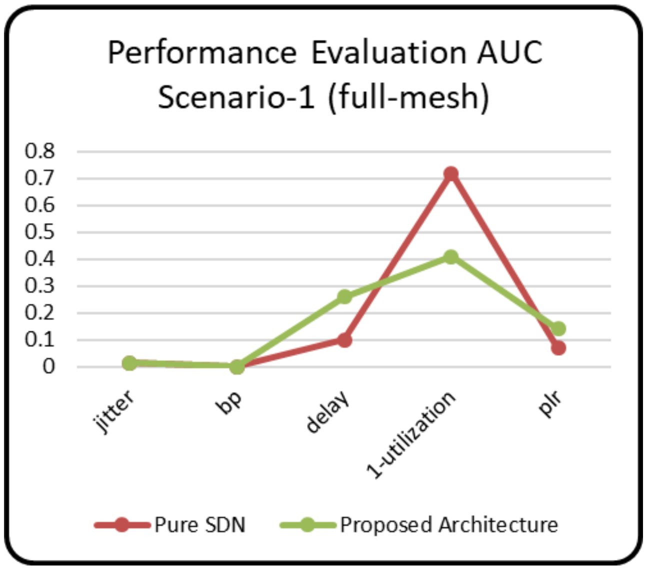

Fig. 7 Average of performance metrics in scenario-1 (full-mesh)

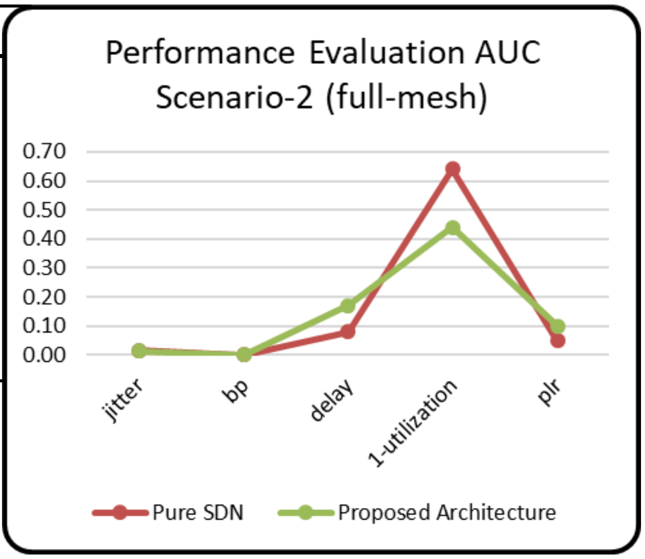

Fig. 8 Average of performance metrics in scenario-2 (partial-mesh)

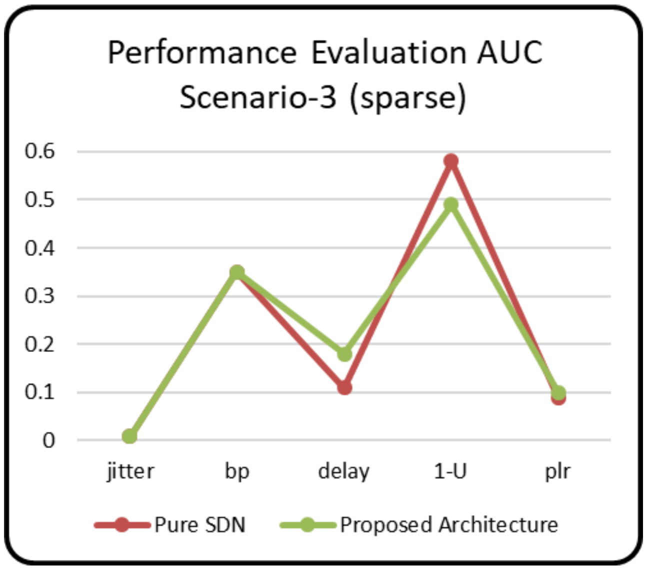

Fig. 9 Average of performance metrics in scenario-3 (sparse)

The area in each scenario is measured and presented in Table 5.

Table 5: Area under the curve for scenario1, 2, and 3

	Scenario-1	Scenario-2	Scenario-3
Autonomic SDN	0.412	0.362	0.564
Pure SDN	0.452	0.392	0.569

The area under curve shows that the proposed architecture has better performance in comparison to the pure SDN.

6- Experiments and Results

SDN is an architecture that makes the network programmable and agile. The autonomic model is a reference model that has been proposed by IBM to make the systems autonomic. In this paper, we designed a new architecture that is composed of SDN and IBM autonomic model. The momentous aim of our research is an SDN-based architecture to control and improve network performance metrics. The performance metrics which have been used in this paper are blocking probability, delay, jitter, packet loss rate and utilization. This problem is an optimization one that should be optimized. Delay, jitter, and packet loss rate should be minimized, but the utilization should be maximized.

Our proposed autonomic SDN architecture has three layers, including the data layer, autonomic control layer, and Learning route learning layer. The data layer is the same as in pure SDN, but the other two layers have been developed in the POX controller with version 0.2.0. Our autonomic controller routes the flows based on the greedy algorithm. The proposed routing algorithm maximizes utilization and minimizes the other performance metrics. The other layer which is route learning collects data in PostgreSQL and analyzes the number of bytes, packets and time to transmission. This layer trains a deep neural network model for flow discrimination. The model updates periodically to make the model more accurate. Our proposed architecture is used in comparison with POX OpenFlow.spanning_tree module to tackle the loop in the network. To evaluate our autonomic controller which makes the performance of the networks improved, we use the area under the curve to show that all 5 metrics are optimized. Three scenarios with 9 nodes are assumed in this research with different numbers of links which are 36, 24, and 9 called full-mesh, partial-mesh, and sparse respectively. The results of simulation show that our proposed SDN architecture has more (minimized) optimized performance in comparison with pure SDN. According to the simulation results, the area under curve difference with pure SDN is increasing with moving from the full-mesh scenario towards the sparse scenario. Our proposed architecture works better in the network with more links, and with the decrease of links the effect of our architecture is declining.

References

[1] J. Pan, S. Paul, and R. Jain, “A survey of the research on future internet architectures,” Commun. Mag. IEEE, vol. 49, no. 7, pp. 26–36, 2011.
[2] H. Qi and K. Li, Software Defined Networking Applications in Distributed Datacenters. Dalian, China: Engineering, SpringerBriefs in Electrical and Computer, 2016.
[3] N. Feamster, J. Rexford, and E. Zegura, “The Road to SDN: An Intellectual History of Programmable Networks,” ACM Sigcomm Comput. Commun., vol. 44, no. 2, pp. 87–98, 2014.
[4] R. Masoudi and A. Ghaffari, “Software defined networks: A survey,” J. Netw. Comput. Appl., vol. 67, no. May, pp. 1–25, 2016.
[5] A. K. Singh, “A survey and classification of controller placement problem in SDN,” Int. J. Netw. Manag., no. December 2017.
[6] S. Bera, S. Misra, and A. V. Vasilakos, “Software-Defined Networking for Internet of Things: A Survey,” IEEE Internet Things J., vol. 4662, pp. 1–1, 2017.
[7] Y. E. Oktian, S. Lee, H. Lee, and J. Lam, “Distributed SDN Controller System: A Survey on Design Choice,” Comput. Networks, vol. 121, pp. 100–111, 2017.
[8] A. Shirmarz and A. Ghaffari, “Performance issues and solutions in SDN-based data center: a survey,” J. Supercomput., 2020.
[9] A. T. Oliveira, B. Jos, C. A. Martins, M. F. Moreno, and A. B. Vieira, “SDN-Based Architecture for Providing QoS to High-Performance Distributed Applications,” in IEEE Symposium on Computers and Communications (ISCC), 2018.
[10] O. Aldhaibani, F. Bouhafs, M. Makay, and A. Raschellà, “An SDN-based Architecture for Smart Handover to Improve QoE in IEEE 802. 11 WLANs,” in 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2014, pp. 287–292.
[11] I. Engineering, “Enhancing the performance of future wireless networks with software-defined networking ∗,” Front. Inf. Technol. Electron. Eng., vol. 17, no. 7, pp. 606–619, 2016.
[12] T. Shozi, P. Mudali, and O. Matthew, “An SDN Solution for Performance Improvement in Dedicated Wide- Area Networks,” in 2019 Conference on Information Communications Technology and Society (ICTAS), pp. 1–6.
[13] J. W. Kleinrouweler, S. Cabrero, and P. Cesar, “Delivering Stable High-Quality Video : An SDN Architecture with DASH Assisting Network Elements Categories and Subject Descriptors,” in Proceedings of the 7th International Conference on Multimedia Systems, 2016.
[14] W. Wendong, Q. I. Qinglei, G. Xiangyang, H. U. Yannan, and Q. U. E. Xirong, “Autonomic QoS Management Mechanism in Software Defined Network,” China Commun., no. July, pp. 13–23, 2014.
[15] G. Poulios, K. Tsagkaris, P. Demestichas, A. Tall, Z. Altman, and C. Destré, “Autonomics and SDN for self-organizing networks,” 11th Int. Symp. Wirel. Commun. Syst. ISWCS 2014 - Proc., pp. 830–835, 2014.
[16] P. Neves et al., “The SELFNET Approach for Autonomic Management in an NFV/SDN Networking Paradigm,” Int. J. Distrib. Sens. Networks, vol. 2016, no. 4, 2016.
[17] IBM, “Autonomic Computing White Paper: An Architectural Blueprint for Autonomic Computing,” IBM White Pap., no. June, p. 34, 2005.
[18] V. W. Protocol, “OpenFlow Switch Specification,” 2012.
[19] C. Zhang and X. Wang, “Deep learning-based network application classification for SDN,” Trans. Emerg. Telecommun. Technol. Wiley Online Libr. J., no. February 2018.
[20] A. Shirmarz and A. Ghaffari, “An adaptive greedy flow routing algorithm for performance improvement in a software-defined network,” Int. Numer. Model. Electron. networks, Devices, Fields-Wiley online Libr., no. March, pp. 1–21, 2019.
[21] T. Auld, A. W. Moore, and S. F. Gull, “Bayesian Neural Networks for Internet Traffic Classification,” IEEE Trans. Neural Networks, vol. 18, no. 1, pp. 223–239, 2007.

Alireza Shirmarz received his BS, MS, and P.hd degree in computer engineering, IT, and Computer system architecture from Tehran Shahed, Tehran Polytechnic, university respectively. He has worked on software defined networking and performance. His research interests include Network, Recommendation systems, AI, Data mining and Data Science.

Ali Ghaffari received his BSc, MSc. and Ph.D. degrees in computer engineering from the University of Tehran and IAU (Islamic Azad University), TEHRAN, IRAN in 1994, 2002 and 2011 respectively. As an associate professor of computer engineering at Islamic Azad University, Tabriz Branch, IRAN, his research interests are mainly in the field of software defined network(SDN), Wireless Sensor Networks (WSNs), Mobile Ad Hoc Networks(MANETs), Vehicular Ad Hoc Networks(VANETs), networks security and Quality of Service (QoS). He has published more than 60 international conference and reviewed journal papers.