International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 8 (2020), pp. 1880-1895
 https://dx.doi.org/10.37624/IJERT/13.8.2020.1880-1895

AI-aided Traffic Differentiated QoS Routing and Dynamic Offloading in Distributed Fragmentation Optimized SDN-IoT

Muhamed Begović[1], Samir Čaušević[1], Belma Memić[1] and Adisa Hasković[1]
[1] *Faculty of Traffic and Communications, University of Sarajevo, Bosnia and Herzegovina*

ORCID: 0000-0002-3691-4969 (Muhamed Begovic)

Abstract

Internet of Things (IoT) becomes an emerging network technology that expedites billions of devices to be connected via the Internet to provide real-time intelligent application services. The benefits of Software-Defined Networking (SDN) can be used to fulfill IoT requirements. Quality of Service provisioning is an on-going demand in software-defined IoT (SD-IoT), particularly for large scale environments. In this paper, we address this issue by proposing a seamless model of AI-aided Traffic Differentiated QoS Routing and Dynamic Offloading in distributed fragmentation optimized SDN-IoT. Firstly, we propose a Multi-Criterion based Deep Packet Inspection method for classifying the network traffic, which is held in Edge Routers (access points). Secondly, we construct a Partially Connected Network Topology using the ISOMAP algorithm for an effective rule placement and routing. We propose a Traffic Differentiated QoS Routing for forwarding data packets via the most suitable switches. We select the optimum route by Deep Alternative Neural Network (DANN). Based on the relationships among switches, the path is selected and flow rules are deployed. The poor QoS is often caused by load imbalance in controllers and switches. To overwhelm this issue, we propose a Dynamic Offloading scheme in SD-IoT. We offload the data packets from the overloaded controller to the underloaded controller using Hassanat Distance-based K-nearest neighbors (HDK-NN) algorithm. Similarly, we propose a Ranking-based Entropy function (R-Ef) to allow dynamic offloading among switches. Simulation is performed using the NS3.26 simulator and the results proved that our proposed AI-aided SD-IoT model provides superior QoS performance compared to previous approaches.

Keywords: Software-Defined Internet of Things, QoS Provisioning, Artificial Intelligence, Traffic Classification and Routing, Dynamic Offloading, Rule Placement

I. INTRODUCTION

Software-Defined Internet of Things (SD-IoT) is a new network management and control technology that supports diverse real-time applications [1-3]. SDN consists of data forwarding and control over the network devices. It separated the control logic from forwarding devices and controls it from the single entity, which is called a controller. SD-IoT is a future Internet technology that supports a wide range of applications such as Industrial IoT, Sensor Networks, and so on [4-8]. Quality of Service (QoS) provisioning is a potential need in SD-IoT (delay-sensitive or loss-sensitive). To improve the QoS while providing control and management of SDN, different mechanisms have been proposed such as routing, queuing theory, scheduling, traffic classification, load balancing, and rule placement. Task offloading is a current topic in SD-IoT applications. However, employing a huge number of heterogeneous IoT devices in a centralized SDN controller does not meet the QoS requirements. Hence, the multi-controller enabled distributed environment is presented. In a distributed environment, the overloading of the individual controller is a core issue [9-13]. The fragmentation method was introduced in [14]. This approach uses two different controllers such as local controllers and the global controller. The global controller has a global view of the local controller's management and control. In software-defined WSN, a local controller is connected with the sink node and it communicates and gathers data from sensor nodes.

A hierarchical control plane for the multi-domain environment is considered in [15]. Preserving topology in network traffic is one of the big objectives of this paper. For this purpose, the local controller is assigned for each domain (geographical area), and the global controller has a global view for monitoring of all local controllers in different domains. However, the load is imbalanced in the network as a consequence of the hierarchical structure. The upper layer of controllers is balanced and the bottom layer is very under-loaded, which has given less priority. It does not update the frequent network changes and the global controller fails to handle failures among local controllers immediately. An optimum load-balanced path is built through network topology and this minimizes new assignments by the best dynamic offloading strategy. Load balanced routing is the way to achieve QoS goals in SDN. Conventional routing algorithms such as the Bellman-Ford algorithm, Link State algorithm, and Dijkstra algorithm are presented to improve balance the load in routers. But these algorithms are time-consuming and not effective for load-balanced routing.

1.1. Motivation

Various meta-heuristic algorithms have been proposed for load balancing in SD-IoT such as ant colony optimization (ACO), simulated annealing (SA), genetic algorithm (GA), particle swarm optimization (PSO), and so on. However, these optimization algorithms do not handle large communication overhead. With the growth rate of network devices and their continuous sensing feature, the centralized controller cannot handle a large number of requests. Hence multi-controller mechanism is introduced currently, which seeks to address the research issues of a single controller. Fig. 1. illustrates the offloading in SDN. In multi-controller SDN, there are three types of controller mechanisms implemented. Control plane can be implemented using a central controller managing the local controllers, distributed controllers that work and take action on their own, and fragmentation-based distributed

controllers. Switch migration is one of the solutions in the multi-controller environment for load balancing [16-17]. In a distributed environment, switch migration is not adequate and it leads to time complexity. Flow rule placement in switches is an emerging research topic in SDN. Recently, it gained more attention among researchers. When the need for the flow rules installation is high, then the flow table of switch might be overloaded, which can cause severe issues in QoS provisioning. Besides, when flow rules are installed based on per-flow statistics, it causes high computation overhead. Therefore, in this paper, we addressed the above-mentioned issues and designed a distributed controller with the fragmentation-optimized environment for QoS provisioning.

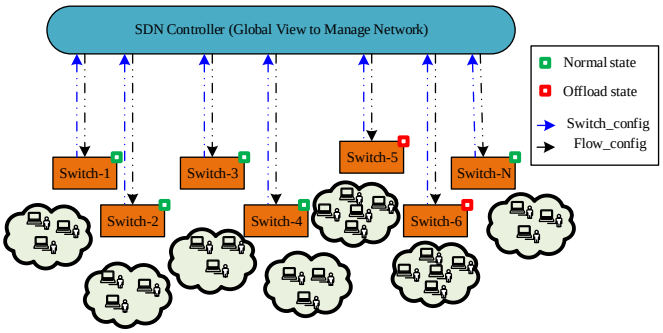


Fig. 1. Offloading in SDN

1.2. Contributions

The objective of this paper is to design a novel artificial intelligence aided SD-IoT to increase the QoS in each layer (devices layer, switches layer, and controller layer). The proposed traffic differentiated routing and dynamic offloading model for QoS improvement are defined on an individual layer. The major contributions of this paper are summarized as follows:

1) For increasing the QoS analysis, the AI-aided SD-IoT model performs four actions: Traffic Differentiation, Switches Topology Discovery, Traffic Differentiated Routing and Rule Placement, and Dynamic Offloading.
2) Traffic differentiation - Initially, we differentiate the traffic arrived from edge routers. Based on the traffic type, appropriate routing action is taken. We propose a Multi-Criterion-based Deep Packet Inspection method. To compute the QoS satisfaction of the IoT user, we introduce the delay tolerance for network traffic arrived on ER.
3) Switches Topology Discovery - ISOMAP algorithm is proposed for topology management, which effectively determines the link connectivity. Partially Connected Topology connects based on four criteria: CPU computing resources, node degree, queue utilization, and link quality. This topology consists of three relationships: one-to-one, one-to-many, many-to-many.
4) Traffic Differentiated Routing - Traffic Differentiated Routing uses a Deep Alternative Neural Network (DANN), which considers several metrics for load-balanced routing. When the flow rule is not matched with the flow table, then the route request is sent to the respective controller. The controller validates the connectivity among switches and then places the rule. It reduces the number of rules employed on the switches.
5) Dynamic Offloading – Our model predicts the switch utilization and packets migration on the controllers. Packets migration is implemented using Ranking based Entropy Function, and Hassanat Distance-based K-Nearest Neighbor (HDK-NN) algorithm is used for switch offloading from the overloaded controller to the underloaded controller.
6) Experiments show that the proposed model achieves 12% of reduced end-to-end delay, 30% of reduced packet loss rate, 35% of reduced switch failure rate, 15% of reduced controller failure rate, 12% of increased throughput, 65% of reduced rule placement and 30% of increased load balancing rate.

1.3. Paper organization

This paper is structured as follows: Section II details the state-of-the-art on the QoS improvement concepts in SD-IoT. Section III presents the problematic issues of previous research. Section IV deals with the proposed AI-aided SD-IoT model. Section V elaborates on the performances of the proposed model and comparison with previous research by significant parameters. In section VI, we conclude the paper and give future directions.

Table 1. Nomenclature

Notation	Meaning
$S_i = (S_1, S_2, S_3\ldots, S_n)$	Switches
$T_{1xN} = [T(1), T(2), T(3),\ldots T(M)]$	Network topology
$D_1, D_2, D_3\ldots, D_n$	Devices
L_{ij}	The link between switch i and j
DC	Dominant Controller
LC_i	Leaf Controller
R^2	Region of switches
lQ	Link Quality
Ð ţ	Delay time
Q(i)	Queue utilization
δ(i)	Switch buffer
r_m	Relative mobility
D_{ij}	Distance
$N(p_i)$	Number of packets
α and β	Coefficient parameters

II. STATE-OF-THE-ART

In this section, we elaborate on the different state-of-the-art from four different perspectives - traffic differentiation, routing, offloading, and rule placement in SDN and IoT-based QoS schemes.

2.1. Traffic differentiation schemes

In IoT, forecasting network traffic is a critical point for SDN. Dias et al. [18] proposed a traffic classification approach for real-time applications (e.g. video streaming application). A naïve Bayes classification method is proposed for real-time video streaming traffic network classification, which primary

intention is to reduce the delay for multimedia service applications requested by the user. User preference is computed for traffic classification and delay tolerance for sensitive applications. The naïve Bayes algorithm fails to classify the incoming traffic when the network traffic is high. Martin et al. [19] proposed a deep learning-based neural network architecture for IoT traffic prediction. This architecture contains residual, boosted, and stacked networks. The proposed neural network architecture shows good results, but the time complexity is high.

Yu et al. [20] proposed a QoS-aware traffic classification method in SDN by DPI. A semi-supervised machine learning algorithm is combined with DPI for SDN. This method classifies the arrived flows into different QoS classes. A neural network-based traffic matrix is constructed for traffic prediction. The authors in [21] use long short-term memory recurrent neural networks (LSTM RNNs). From the current history of traffic, future network traffic is predicted. LSTM and RNN consume significant time for traffic prediction and require substantial processing time. Tajiki et al. [22] have considered congestion control and joint QoS for traffic prediction in SDN. This paper solves two optimization problems including an exact solution and fast suboptimum one. The authors consider the bandwidth and delay constraints of SDN. In routing, modules for resource allocation and resource re-allocation are used. The rerouting module consumes time which is not negligible.

2.2. Routing schemes

Park et al. [23] presented a network situation-aware framework (NSAF) for handling application routing in SDN. The route must be based on the QoS requirements and dynamic network status changing. It consists of application registration, network status monitoring, violation detection, and routing control. For different service classes (application type), different serv class is incorporated such as packet loss, delay, and jitter. It does not manage the dynamic network changes and control paths when application requirements change. Further, NSAF uses Dijkstra's routing and genetic algorithms in routing, which induces high computational overhead. Saha et al. [24] proposed a traffic-aware QoS routing in SD-IoT networks. In general, the route is constructed for two types of applications: delay-sensitive and loss-sensitive. A greedy approach-based K-shortest paths algorithm is proposed to compute the optimum routing path in which QoS requirements are considered for each packet. In this step, the controller is deployed with adaptive flow rules for routing switches. Moreover, IoT users do not only request delay-sensitive and loss-sensitive applications. In a large-scale network, different heterogeneous devices with various requirements are present.

A simulated annealing based QoS-aware routing (SAQR) algorithm is proposed in [25]. It adaptively adjusts the weight of delay, loss rate, and bandwidth requirements to determine the best routing path with meeting the QoS requirements. Experiments were conducted to validate this approach concerning loss rate, delay, and bandwidth. Also, authors select the best path using Dijkstra's algorithm, which is a blind search algorithm that consumes significant time for processing.

Routing is a compelling solution for balancing the load and overhead in the network. Low-cost load balancing route management (L2RM) is an effective framework proposed by Wang et al. [26]. Adaptive route modification is implemented to avoid flow table overloading. Further, L2RM uses dynamic information polling (DIP) scheme, which queries switches to know the current queue utilization. When failure occurs, the response time of the controller increases and it leads to poor network management. Another similar research can be found in [27] and it is called load-balanced aware routing on the SDN controller. It addresses the problem of load balance routing in both controllers and links and thus it minimizes the controller response time and link utilization rate. Two update mechanisms are proposed such as area bound update and controller load update. Experiment results show that the controller response time is greatly reduced by balancing the load, but it fails to minimize the overhead among switches.

2.3. Offloading schemes

SDN controller is fully capable of offloading the tasks dynamically. Detour [28] proposes a dynamic allocation of tasks and resources in software-defined Fog networks for IoT applications. IoT devices are connected to fog nodes using multi-hop IoT APs. The SDN controller collects network information through the southbound interface and performs optimum task allocation thanks to the global view of the network. The idea is to decide whether the task will be performed locally or on a remote device, select the ideal fog device, and select the optimum path to forward the task to another device. The M/M/1 model is applied to the task queue to select the appropriate application to perform the task after reaching the fog node. The limitation is that the end-to-end delay is significant in terms of sending load balancing requests to the SDN controller, and high-priority tasks have to wait a long time. Neghabi et al. [29] presented the solutions for load balancing using Meta-Heuristic algorithms. The authors have presented some benefits of using these algorithms for improving network performance. However, these optimization algorithms do not solve the problems of large network loads and most of them face the problem of falling into the local optimum and premature convergence.

Authors in [30] have considered a load-balancing issue by deploying a virtual SDN controller (VController). When network traffic is high, the virtual SDN controller is deployed over the network. For this purpose, a virtual network function (VNF) is used with primary and secondary VControllers. When the load of the primary controller increases, the secondary controller splits the load and processes the part of requests. The second controller has a copy of the primary controller and it balances the load among switches. A large number of load balancing actions are required and poor QoS is achieved due to virtual controller placement. In Machine to Machine (M2M) networks, traffic-aware load balancing is proposed [31]. It is implemented in SDN assisted IoT. The processes involved in this paper are: (1) determining traffic flow at arrived switches using packet header information, (2) the route is considered and the determined traffic flow is forwarded and processed via the route, and (3) flow table is updated if the latency exceeds the threshold value. For traffic-aware load balancing, only delay and type of service are considered which is not sufficient for delay-tolerant applications (M2M services).

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 8 (2020), pp. 1880-1895
 https://dx.doi.org/10.37624/IJERT/13.8.2020.1880-1895

A distributed multi-controller environment-based switch migration is proposed in [32]. The authors proposed the strategy called efficiency aware switch migration (EASM) for distributed controllers load balancing. The load different matrix and trigger factor are used to estimate the controllers' load balancing. The migration efficiency problem considered load balancing rate and migration cost for optimum migration of switches. EASM is not suitable for switch migration in a large scale network environment.

Authors in [33] have focused on dynamic load balancing with hybrid genetic and ant colony optimization algorithms. In GA, the fitness value is computed by path length, energy consumption rate for packets transmission and reception, and energy status of all switches. In ACO, the optimum path is identified and the packet is forwarded towards the selected path. Search speed for path selection is sufficient, but criteria for path selection are insufficient to find the real optimum path.

In [34] authors proposed a predictive adaptive real-time model that is used to select cloud for random service prediction available in the virtual network services over a multi-cloud environment. Different types of cloud provide different services with different QoS. For this purpose, dynamic virtual placement was applied in this study. With this dynamic virtual placement, different kinds of traffic were served for users from a different region. When arrived users' traffic is reached from the expected level, then more virtual placement is required. Hence, dynamic offloading among switches will be helpful in this case.

2.4. Rule placement schemes

Flowstat [35] presents flow-rule placement considering three processes, namely route selection, rule installation, and rule redistribution. The authors formulated the Max-Flow-Min-Cost optimization problem for optimum forwarding paths selection. For the computed path, flow rules are installed. Furthermore, rule distribution is considered for flow rules traffic congestion in the network, which decreases the network traffic. It minimizes the end-to-end delay for packet forwarding and flows rule installation in the data plane. When the path is frequently selected, the installation of flow rules for the particular path is high and it causes flow table overloading in switches. The controller does not consider the adequate metrics for forwarding path selection, which causes severe issues in QoS improvement.

Rule placement is a challenging issue due to the memory limitation of the switches. A novel approach is proposed [36] for rule placement according to two possible connections between neighboring switches: serial and parallel relationship. For rule placement handling, in this paper, a new data structure called OPTree (Ordered Predicate Tree) is proposed. It is used to represent rules in the device and is suitable for checking if a rule is contained in an existing rule. Two OPTree algorithms have been created for setting up rules and searching. Although the approach presented considers the positional relationship for adjacent devices, the number of rules being set is still large, which places a heavy burden on the controller. The use of fat-tree or star topologies increases the number of rules that are added, making network topology an important factor for global network management. In [37], the SDN controller computes and places new rules for switches flow table. If a switch receives the data packet, but it does not match with the flow table rules, the switch directly communicates to the controller. In this paper, the controller places the flow rule for a particular switch and also gives the best path for processing a packet. Then the packet is transmitted to neighboring switches. For flow rule installation, switch-controller delay and switch-to-switch delay is computed for packet transmission via the path. Computation of path delay between switch–controller and two switches is relatively high. Traffic overhead is increased and the flow table of the switch is overloaded within a few packets processing.

III. PROBLEM DESCRIPTION

Based on the previous state-of-the-art analysis, in this section, we defined the problems on the four concepts focused on QoS enhancement. Fig. 2. presents the problem statement undertaken in the AI-aided SD-IoT model. Further, we address the following research questions.

- (RQ1). How our AI-aided SD-IoT model can efficiently process a large number of requests from IoT devices?
- (RQ2). What are the performance benefits of proposing a new model in SD-IoT with subject to QoS provisioning?
- (RQ3). Is it best for forwarding large flow requests from the multi-controller environment?
- (RQ4). How AI-aided SD-IoT model is best for resource-constrained IoT devices?

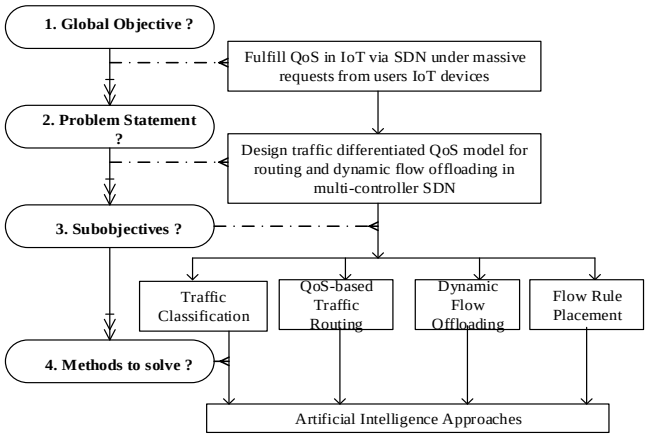


Fig. 2. Problem statement representation

In the previous section, several heuristic optimization algorithms are illustrated. These algorithms produce large communication overhead since they can provide a single optimum solution and poor scalability. For SD-IoT based networks, a large AI model is required, which eliminates the communication overhead, packet loss, and latency issues. Also, the AI model is needed for environments with a large amount of traffic and providing results within the expected delay.

In [38] authors have contributed with three operations including network dimension reduction by Principal Component Analysis (PCA), Queue Utilization (QU) prediction, and load-balanced routing by Deep Neural Network (DNN). In the local router, packets arrived status is monitored and the next hop is selected for transmission whereas the central router is used to detect the QU and traffic rate of all local routers. This model has several drawbacks as follows: it is a time-consuming operation, it does not produce the meaningful

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 8 (2020), pp. 1880-1895
 https://dx.doi.org/10.37624/IJERT/13.8.2020.1880-1895

network patterns, the packet loss rate is high because it does not consider the link quality, extracting network topology is a tedious task, and it does not result with effective SDR representation. Load balanced routing for next-hop selection is based on QU which is predicted by DNN, which is not sufficient to achieve load balancing. A task with high priority (real-time) is required to wait for a longer time. Furthermore, SDR is not suited for a large-scale environment because the central router would lead to a single point of failure.

To avoid the frequent flow migrations from one controller to another, fractional level switch migration is proposed in [39] by the dynamic controller mapping algorithm. There are 3 steps incorporated: (1) load imbalance detection, (2) target controllers and switches selection, and (3) switch migration (target controller load is balanced). Subsequently, controllers are reordered. In a large multi-controller environment, search spaces for finding the target controllers are difficult and the solution can be suboptimum. To minimize the number of new assignments, an absolute prediction of switches load is significant to consider.

A new routing strategy is proposed in [40], which is called Segmented Routing. A segment refers to Instruction and a node segment consists of a unique label of next switch to reach. Initially, the Multi-Objective Particle Swarm Optimization algorithm is presented for link weight optimization to load balancing and path cost. With the use of optimized Weighted_Matrix between the source node and the destination node, K-number of shortest paths is selected and the best path is selected based on user preferences. For user preferences, QoS is evaluated using the Key Performance Index (Delay, Jitter, and Packet Loss). The KPI is computed by the Fuzzy AHP algorithm (Analytic Hierarchy Process). Segment routing using the MOPSO algorithm is a very time-consuming task, and also Fuzzy AHP for weight matrix computation is not effective since it is a relatively complex method and it requires a large size of mathematical computations in measuring priority. To deal with the load balancing issue, the arrival packet rate is one of the most important metrics, which must be considered for load-balanced routing. The authors of this paper have not concentrated on this issue.

In this paper, we concentrated on the core issues to upgrade the QoS of the network. These problems are solved by the proposed solutions in this paper and are elaborated in the next section.

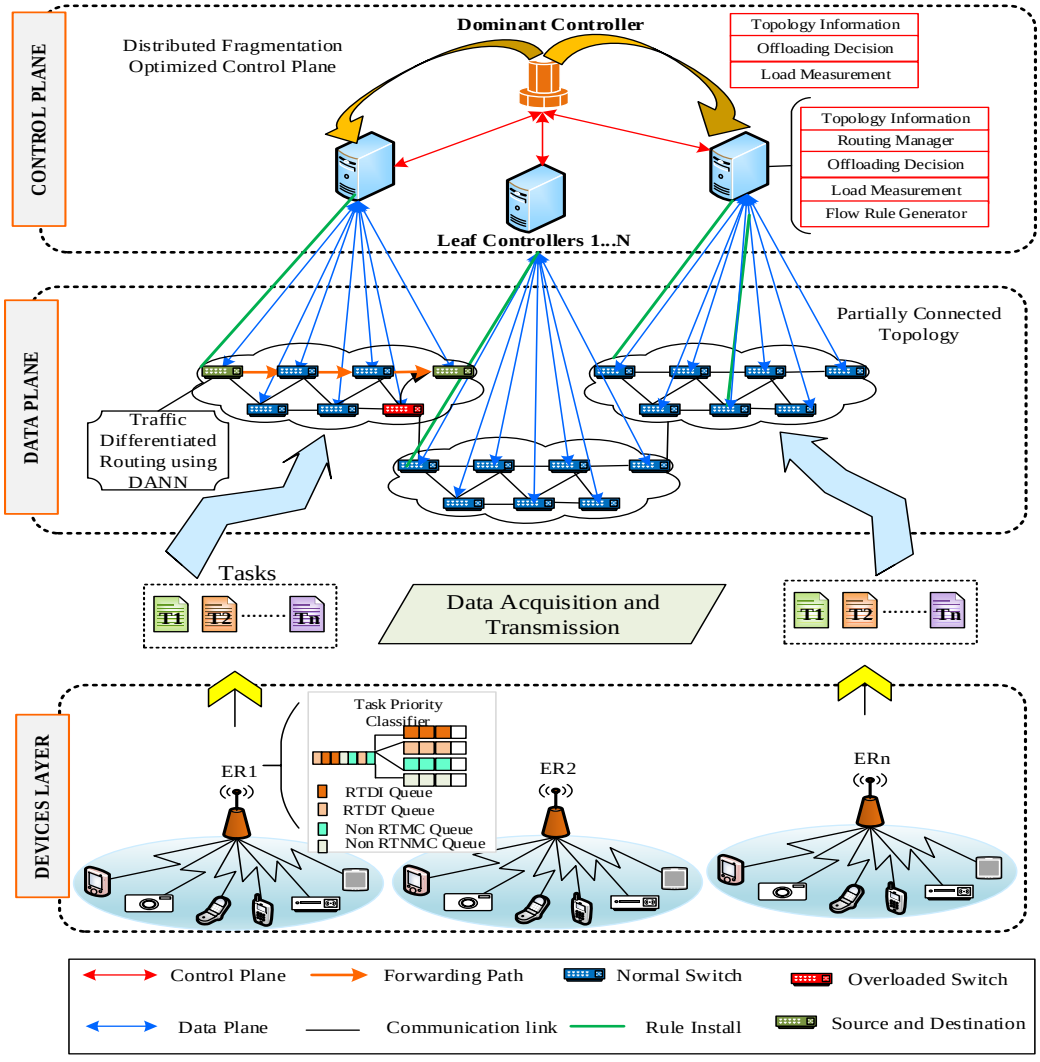


Fig. 3. System model

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 8 (2020), pp. 1880-1895
 https://dx.doi.org/10.37624/IJERT/13.8.2020.1880-1895

IV. AI-AIDED SOFTWARE-DEFINED IOT

In this section, we present the system model of the AI-aided SD-IoT model. The notations used in this paper are listed in Table 1.

4.1. System overview

In this paper, we assume that our proposed model for SD-IoT is employed with heterogeneous IoT devices constrained in resources. They are connected and communicating via the Internet over the same IP assisted gateways, which are called Edge Routers (ERs). Our proposed work consists of distributed controllers including Dominant Controller and Leaf Controllers, Switches, ERs, and IoT devices. Fig. 3. shows the AI-aided SD-IoT architecture. Assume that SD-IoT environment is an undirected graph $\mathcal{G} = (\mathbb{S}, \mathbb{L}, \mathbb{c})$, where S represents the set of all switches, L represents the links between switch i and j, and c is a controller. The link between switches are:

$$\mathbb{L} = \{(i,j) | (i,j) \in s \times s, i \neq j\} \quad (1)$$

As mentioned earlier, links between switches are a key point in the SDN. The DC communicates with the LCs and LCs communicate with switches via OpenFlow protocol.
The communication between the SDN controller and the application layer is obtained via Application Programming Interface (API). There are three layers incorporated for designing our proposed model:

- Devices Layer: In this layer, several heterogeneous IoT devices are $D_1 \ldots D_n$. From device D_i, the request is sent to the ER, which performs the traffic differentiation by multi-criterion based deep packet inspection.
- Data Plane Layer: In this layer, several switches are deployed in a partially connected network topology. Switches verify the flow rule and take the action based on the flow table information.
- Control Plane layer: This layer consists of fragmentation optimized distributed controllers. It reduces the problems caused by the centralized and distributed controller environments.

There are four operations involved in our approach: Traffic Differentiation, Topology Discovery, Traffic Differentiated Routing and Rule Placement, and Dynamic Offloading. In the following sub-sections, we elaborate on these operations.

4.2. Traffic differentiation

In the first layer, IoT devices are deployed and sensing is started. The sensed information is acquired and transferred via ERs. In the ER, traffic is classified into four classes and put into individual queues:

(1). Real-Time Delay Intolerant (RTDI) Queue
(2). Real-Time Delay Tolerant (RTDT) Queue
(3). Non-Real-Time Mission Critical (Non-RTMC) Queue
(4). Non-Real-Time Non-Mission Critical (Non-RTNMC) Queue

Traffic from IoT devices is differentiated (see Fig. 4.) to meet the QoS requirements.

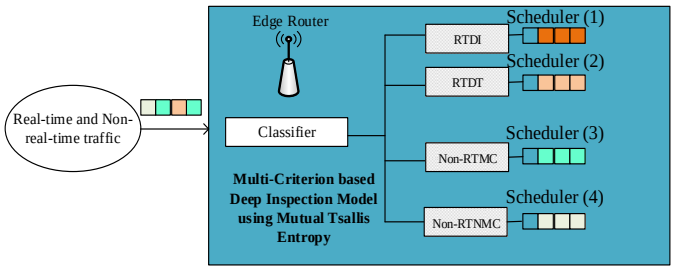


Fig. 4. Traffic differentiation

Some of the applications are video streaming (YouTube, Dailymotion, and Netflix), P2P torrent (BitTorrent, and VUZE), video chat or VoIP (Gtalk, Facebook Messenger, Skype, WhatsApp, Twitter), cloud access (AWS, Google Drive, OneDrive, and Dropbox), online games, email sending or receiving. This traffic classification is suited for various application examples. Each IoT device's traffic request is measured in bps. In ER, we have used Deep Packet Inspector and Traffic Classifier. We split the input traffic into four classes using four criteria: (1). Packet Deadline $P_D(i)$, (2). Packet Size $P_S(i)$, (3). Protocol Used $PR_T(i)$, and (4). Type of Service Request $T_{SR}(i)$.

Table 2. Traffic differentiation

Traffic Class	Example service	Required data rate	Packet Delay	Priority
RTDI	video conferencing, and VoIP	1.5-8Mbps	<50ms	1
RTDT	Video or Audio	2-10Mbps	<200ms	2
Non-RTMC	E-commerce	1.5-150Mbps	<500ms	3
Non-RTNMC	Email or Web	Kbps-Gbps	<1000ms	4

Algorithm 1: Traffic Differentiation

Begin
Input: n packets arriving at the ER δ
Output: Four traffic classes (RTDI =1, RTDT =2, Non-RTMC=3, and Non-RTNMC =4)
// Apply Mutual Tsallis Entropy computation for all $p_{(i)}$
for each packet $p_{(i)}$ **do**
If $p_{(i)} \in$ RTDI_level **then**
Put $p_{(i)}$ into RTDI;
else
If $p_{(i)} \in RTDT$_level **then**
Put $p_{(i)}$ into RTDT;
else
If $p_{(i)} \in Non-RTMC$_level **then**
Put $p_{(i)}$into Non-RTMC;
else
Put $p_{(i)}$into Non-RTNMC;
end if
end if
end for

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 8 (2020), pp. 1880-1895
 https://dx.doi.org/10.37624/IJERT/13.8.2020.1880-1895

In deep packet inspectors, these four criteria are evaluated and forwarded to the traffic classifier with the inspected result. Table 2. depicts traffic differentiation in detail.
Based on the packet delay and the required data rate, traffic is differentiated in traffic classifier and priority is given. Algorithm 1 shows the procedure for traffic differentiation. When the arriving traffic is RTDI, then the packet is immediately forward to the switches for taking an action. For packet (traffic) classification, our proposed multi-criterion based deep inspection model follows Mutual Tsallis Entropy function. The Mutual Tsallis Entropy function is defined by δ>1 and it is computed according to the equation:

$$M\varepsilon_\delta^t = h_\delta^t(x) - h_\delta^t(x|y) = h_\delta^t(y) - h_\delta^t(y|x) = h_\delta^t(x) + h_\delta^t(y) - h_\delta^t(x,y), \quad (2)$$

$h_\delta^t(x,y)$ is an upper bound value that has positive and symmetric Tsallis joint entropy. It shows the correlation between two criterions x and y. After that normalized Tsallis mutual entropy is:

$$NM\varepsilon_\delta^t(x,y) = \frac{M\varepsilon_\delta^t(x;y)}{h_\delta^t(x,y)} \quad (3)$$

The entropy values are taken in the interval of 0 and 1. When the value is 0 then x and y are independent, and also $\delta = 1$. The value 1 is taken if and only if $x = y$. In this way, mutual information-based entropy values are computed for all criteria used for traffic classification.

4.3. Switches topology discovery

In the Data Plane, we firstly construct network topology for switches deployed in the network. We presented a Partially Connected Topology. For routing, we extract topological information. For topology construction, the ISOMAP algorithm is used to analyze the topology connectivity. It defines each switch connection to neighbor switches via its Nearest Euclidean Neighbors. ISOMAP algorithm takes as input the distances $D(s_i, s_j)$ between all pairs (i, j) from N data points in the high-dimensional input space X.
ISOMAP is a manifold learning algorithm, which performs better than PCA, and other dimensionality reduction algorithms. It is easy to discover the low-dimension spatial structure in data. The key advantage of Isomap for topology design is that it uses the Geodesic distance to estimate the dissimilarity between two switches and it generates relatively high accurate neighbor switches according to the Error-Prone Distance Information.
Assume that the SDN consists of M switches $\{s_1, \dots, s_m\}$ deployed in a 2D space. Let $s_i \in \mathbb{R}^2$ denote the location of the switch s_i. Without loss of generality, we suppose that the first n switches are considered for topology design. We use Euclidean distance metric as a domain-specific metric for similarity (distance) computation and the distance between every pair of switches is computed as:

$$D(s_i, s_j) = \left(\sum_{k=1}^{D}(s_{ik} - s_{jk})^2\right)^2 \quad (4)$$

Where $D = 2$ for 2D space and similarly $D = 3$ for 3D space. When compared to PCA, ISOMAP preserves the Intrinsic Geometry of switches location since it captures the Geodesic Manifold distances between all pairs of switches. We consider the following attributes for switches:

(1). CPU computing resources: It is the basic and significant component in a switch, and it contains the primary entities such as Buffer, Operator, and Controller. It receives the instructions and performs action and processes requests. If the CPU of the switch is sufficient, then it can process a large number of requests.

(2). Node degree: It is the direct indicator that reflects the node centrality in the network topology. The switch with a higher degree is selected since it can process more requests.

(3). Queue Utilization (QU): It represents the available space in the switch buffer which reflects its ability to keep and store the packets. As for path selection, the higher switch buffer is selected to reduce the packet loss rate in a high network traffic scenario. However, switches must be underloaded, so we properly select the switch i for intermediate node. It is computed as:

$$Q(i) = \frac{N(p_i)\ in\ \mathfrak{d}\ (i)}{Size\ of\ \mathfrak{d}\ (i)} \quad (5)$$

Where $N(p_i)\ in\ \mathfrak{d}\ (i)$ represents the number of packets in the buffer of the switch i.

(4). Link Quality: This metric describes the relationship between the switches concerning the packet received rate p_Υ and packet transmission rate p_{tr}, and hence link quality $\mathcal{G}\ell$ is computed as:

$$\mathcal{G}\ell = \alpha * p_\Upsilon + \beta * p_{tr} \quad (6)$$

Where α and β are the coefficient parameters for computing the link quality that ranges from 0 to 1. With this value, link quality is computed differently.

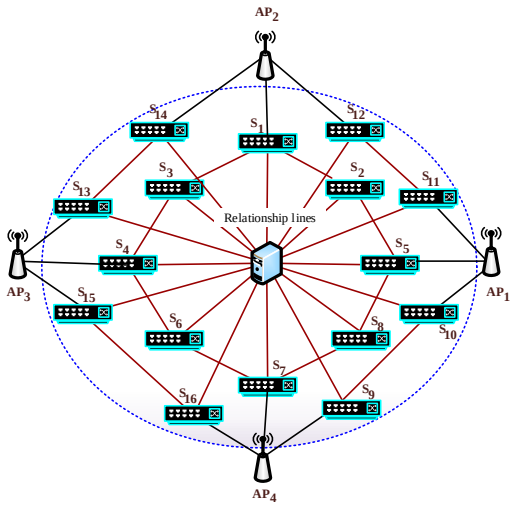


Fig. 5. Partially connected topology

Fig. 5. depicts the connectivity of switches. Therefore, the network topology $T_{1\times N} = [T(1), T(2), \dots T(M)]$ can represent connectivity and availability of switches for choosing the

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 8 (2020), pp. 1880-1895
 https://dx.doi.org/10.37624/IJERT/13.8.2020.1880-1895

particular node for path p. We calculate the parallel and serial relationships between the switches s_i and s_j . A switch containing higher connectivity (serial and parallel) is selected as the intermediate node for packet transmission. Based on the connectivity, LC_i deploys the rules in switches.

4.4. Traffic differentiated routing and rule placement

After network construction, we allow data packets from the queue. Firstly, we process packets from the RTDI queue since they have absolute priority (delay<50ms). Towards that, we process other queues based on the delay tolerance for application services. When a data packet arrives into a queue, we must check the flow rule for the corresponding packet in a switch. If the flow rule is matched in the flow table, then the packet is processed. For that packet, we establish the route. Based on the packet traffic type we construct the route. Hence, we called it Traffic-Aware Routing.

To find the optimum path, several criteria are considered: Packet Loss p_l, RTDI Packets Queue Utilization Rate Q_u, Latency L, Bandwidth B, Hop Count H, and Distance between two Switches D. For traffic-aware routing, we propose a Deep Alternative Neural Network (DANN), which provides better performance than DNN and conventional machine learning methods.

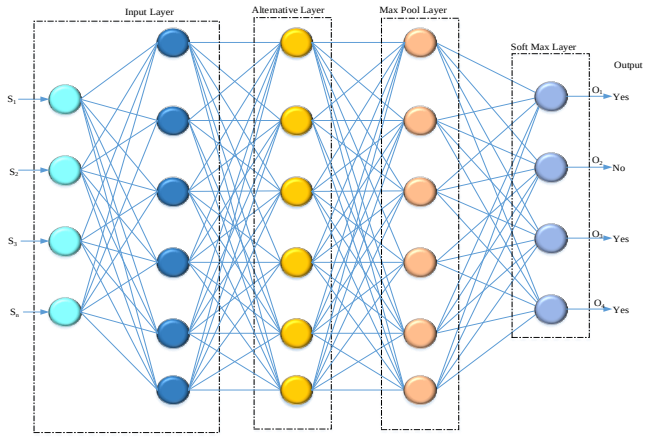


Fig. 6. DANN structure

Table 3. DANN hyperparameters

Layers	Description
6-Alternative layers	64, 128, 256, 256, 512, 512 kernel response maps
Activation function	ReLU
3-Fully connected layers	Size = 2048
Layer kernel size	3× 3 × 3
1 volumetric max pool layer	2× 2 × 2
Learning rate	0.1
Input neurons	512
Batch size	100
Data Samples	1000
Window size	5

DANN is a combination method, which contains the strengths of CNN and RNN. It is depicted in Fig. 6. In DANN architecture, we deal with the alternative layers. Each alternative layer is a convolutional layer followed by the recurrent layer. With the use of Adam Optimizer, DANN parameters are selected and optimized (see Table 3.). To perform DANN training and testing, operations Pyramid Generation and Feature Map Combination were used. It is explored in alternative layers. The recurrent layer in the alternative layer has provided merits such as every element incorporating contexts in an arbitrarily large region in the current layer. When the time steps increase, then the state of every unit is influenced by the other elements that are also increasing, and it is a larger neighborhood in the current layer. If the flow rule does not match with the flow table, then the switch requests the path from the corresponding leaf controller. Then the controller gives the path. With the partially connected topology (parallel and serial relationships to switches), the switch has verified whether the flow rule is present for all switches in the available path. If the rule for a data packet is present in all switches in the available path, then no Rule Placement is required. Otherwise, flow rules are deployed for switches. In this way, rule placement is invoked. The flow table entries according to the traffic class are illustrated in Table 2.

4.5. Dynamic offloading

Offloading refers to offloading the various computational expensive flow requests to the underloaded nodes based on their current load. In this paper, we applied a dynamic offloading scheme on both the data plane and the control plane. Most of the research is focused on balancing the load in either switches or controllers, which degrades the system performance. Hence, in this paper, we propose the concept of dynamic offloading on both the data plane and the control plane at the same time as described below.

4.5.1. Offloading in the data plane

In the data plane, packets are offloaded from the overloaded switch to the underloaded switches based on the current load of the switches. It is called optimum relays selection. For dynamic offloading, we proposed a ranking-based entropy function. This process is held in the leaf controller. We consider several criteria for packets offloading such as packet arrival rate p_{ar}, no. of packets in buffer np_{bf}, the packet loss rate p_{lr}, expected delay ε_d, no. of flow entries n_{fe}, and total no. of packets correctly processed in history $\sum_{i=1}^{n} p(h)$. We compute the entropy value $E_{(i)}$ for all switches in the network for offloading overloaded switch packets into under loaded switches. An attribute set C is used for switches selection:

$$\mathsf{C}= \{p_{ar}, np_{bf}, p_{lr}, \varepsilon_d, n_{fe}, \sum_{i=1}^{n} p(h)\} \quad (7)$$

The mathematical formula to calculate $E_{(p(i))}$ can be defined as:

$$E_{(p(i))} = -\sum_{i=1}^{k} \rho_{(i)} \log_2 \rho(i) \quad (8)$$

Where $\rho(i)$ denotes the probability of switch i being selected in the controller for offload. Information entropy function is used here to address the uncertainty issue. The minimum entropy represents the switch has lower uncertainty and the

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 8 (2020), pp. 1880-1895
 https://dx.doi.org/10.37624/IJERT/13.8.2020.1880-1895

expected entropy is achieved by the higher uncertainty. For all switches, we compute the expected entropy value according to the attribute set C to g, (conditional Entropy), which is computed as:

$$H(g,|C) = \sum_{i=1}^{n} \rho(g,_i)\ H(g,_i) \quad (9)$$

Now, we determine the Information Gain G between the entropy and expectation values as follows:

$$G\ (g,, C) = H(g,) - H(g,|C) \quad (10)$$

Then we compute GainRatio F for each switch $S_i \in LC_i$ according to the attribute set C and select the switches with maximum F.

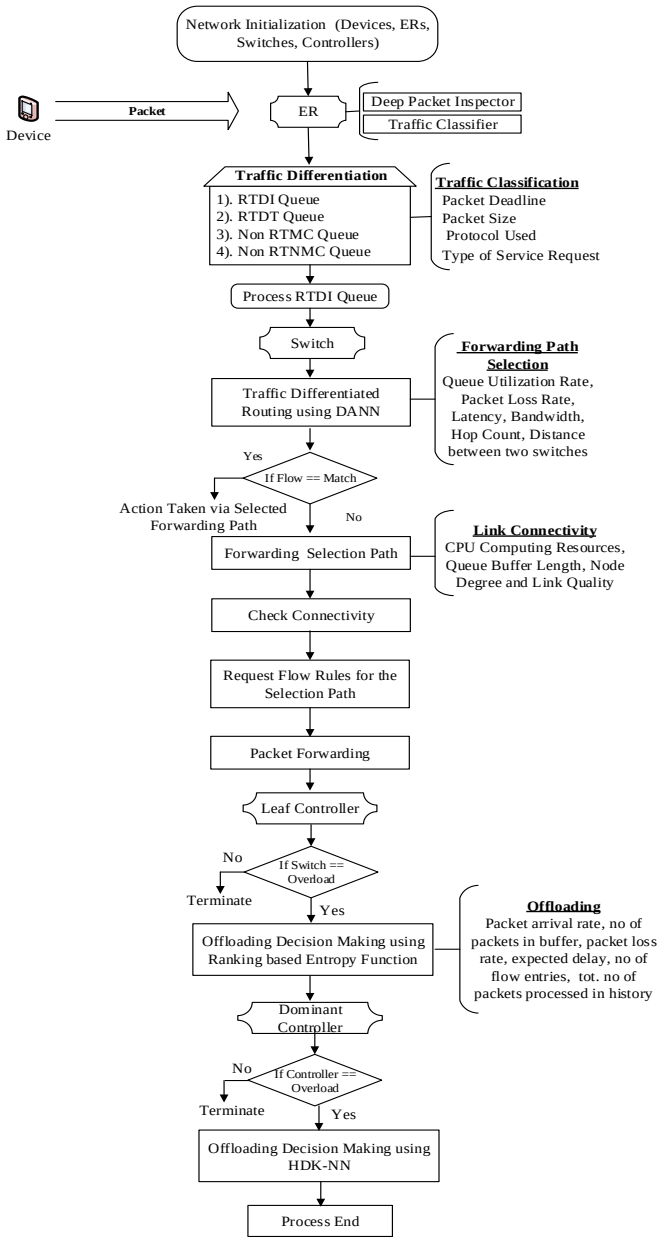


Fig. 7. Workflow diagram

4.5.2. Offloading in controller plane

In this section, we define the controller overloading in the controller plane. Controller load measurement is done in the DC to report the load statistics for all $LC_{(i)}$. The flow statistics include the number of flow table entries $N(\zeta_e)$, the average packet arrival rate Ap_r of each switch i, the round-trip time $\mathbb{R}\mathcal{T}_t$, and the current load $LC_{(i)}$.

The current load of LC is calculated by the number of packets waiting to be processed and currently running on the switches at time t:

$$LC_{(i)} = \frac{\sum_{i=1}^{n} S_i(\mathcal{Y})}{\mathcal{Y}} \quad (11)$$

Where $S_{i=1\ldots n}$ denotes the number of switches, and $S_i(\mathcal{Y})$ is the number of packets from switch i. Each entity's purpose is depicted in Fig. 7. Also, we compute $LoadRatio$ for all $LC_{(i)}$ by the abovementioned factors:

$$LoadRatio = aN(\zeta_e) + bAp_r + c\mathbb{R}\mathcal{T}_t + dLC_{(i)} \quad (12)$$

In equation (12), a, b, c, d are the coefficient weights and their sum is equal to 1.0. Based on the $LoadRatio$, DC determines the overloaded $LC_{(i)}$ and switches connected to it.

When compared to other offloading approaches in SDN, our proposed solution is better in three perspectives:

- Consistency and completeness: The network is full-fledged and it is not discontinuing its services at any time. When the offloading action is taken, then the consistency or stability of switches is high and offloading action is not required for a longer time.
- Overload: In this stage, we compute the amount of overload and overhead in the data plane and controller plane imposed on the system.
- Scalability: The performance is increased and not affected when the number of IoT devices increases.

V. RESULTS

This section contains the evaluation of the AI-aided SD-IoT model with a description of experiments and simulation analysis. In the following subsections, the simulation setup, comparison results, and motivating application are given.

5.1. Experimental setup

To evaluate the QoS for the proposed AI-aided SD-IoT model, we used NS3.26. It is a discrete event network simulator written by C++ programming language, and python scripts are used. Table 5. describes the simulation parameters that are considered for the simulation. Our simulation testbed consists of 50 IoT devices, 6 ERs, 21 OpenvSwitches, 4 Open Daylight Controllers (3 leaf controllers, and 1 dominant controller).

The simulated topology is partially connected topology to connect n switches to the controller. Each switch is connected to a single device and reactivity of switches is investigated in the controller and the status of partially connected topology is updated.

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 8 (2020), pp. 1880-1895
 https://dx.doi.org/10.37624/IJERT/13.8.2020.1880-1895

Further, simulation class libraries were employed to perform the simulation. Our three-layered simulation topology is given in Fig. 8.

Tables 4. and 5. represent the configuration of the system, testbed, and simulation settings respectively. In Table 6., C1, C2, and C3 represent the workload of local controllers 1, 2, and 3 respectively and DC is the dominant controller workload.

Table 4. Configuration of system and testbed

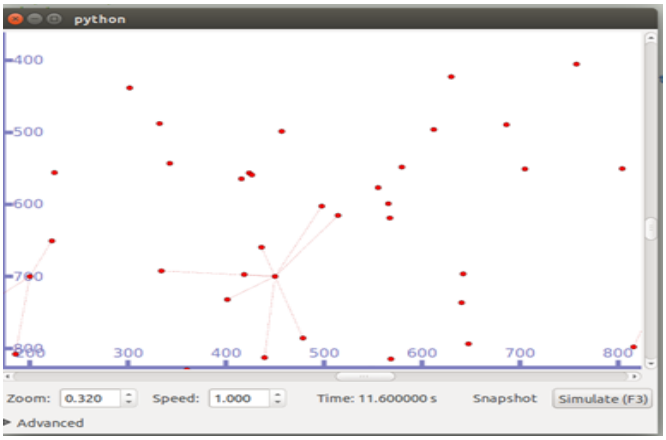
System Configuration	Operating System	Ubuntu 14.04 LTS (32bit)
	Implementation Tool	NS3.26
	Processor	Dual-core and above
	RAM	2GB and above
Testbed Configuration	IoT Devices (Hosts)	3.3GHz, 4-cores, 4GB RAM
	OpenvSwitch	3.3GHz, 4-cores, 4GB RAM
	Open Daylight	3.3GHz, 4-cores, 4GB RAM

Table 5. Simulation settings

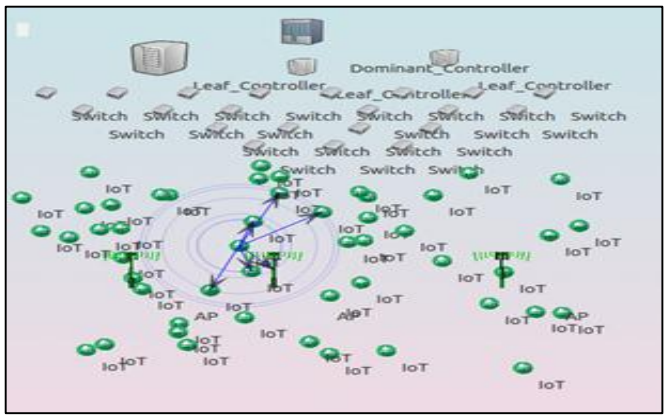
Simulation Parameters		Values
Communication range		1000×1000m
IoT Devices	No of devices	50
	Topology	Partially Connected Topology
	IoT application size	4000-100000MI
	Trained applications	50 per minute
	Devices type	Hosts
	Application type	Real-time and non-real-time
	No of ERs	6
SDN	No of controllers	4
	No of switches	21
	Connection Speed	1000Mbps
	Flow table size	1000 entries
	Interval for flow request	0.001s
	Flow size	100bytes
	Learning rate	0.025
	Switch degree	minimum 3
	CPU of switch	5 units (packets per second)
	buffer length	8
	Start load	100 applications/sec
Traffic Type		TCP, UDP, IP, HTTP, RTP
Packet Size		512 bytes
Number of packets		1,00,000
Packet Interval		50ms
Simulation time		300s
Training Data Samples		10000
Testing Data Samples		5000

Table 6. Controller load statistics

Workload	Value
Maximum workload (C1)	1000 (events/sec)
Maximum workload (C2)	1100 (events/sec)
Maximum workload (C3)	1200 (events/sec)
DC maximum work load	5000 (events/sec)



(a)



(b)

Fig. 8. Simulation testbed: (a) Simulation diagram; (b) Simulation in NetAnim

5.2. Performance metrics

In this section, we define the significant QoS metrics with the necessary formulas considered in this paper for performance evaluation and validation.

(1). End-to-End Delay – It is a time duration caused by the transmission of the packet from the source to the destination. The achievement of QoS leads to the minimization of end-to-end delay. It is computed by:

$$End - to - End\ Delay = \frac{\sum_{p(i)} delay}{number\ of\ packets} \quad (13)$$

(2). Packet Loss Rate – It is a significant phenomenon that leads to the loss of packets traveling from the source to the destination node. The main reason for packet loss is the queue overflow in switches. It is computed as:

$$Packet\ Loss\ Rate = \frac{N(Lpc)\ at\ s(i)}{sum\ of\ packets\ arrived\ at\ s(i)} \quad (14)$$

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 8 (2020), pp. 1880-1895
 https://dx.doi.org/10.37624/IJERT/13.8.2020.1880-1895

Where $N(Lpc)$ represents the number of lost packet count in switch i.

(3). Switch Failure Rate – It is defined as the number of packets failed on switch i at time ţ.

(4). Controller Failure Rate – It is defined as the number of switches failed at the controller at time t.

(5). Throughput – It is defined as the increased packets transmission and reception status of all IoT devices in the network. However, it is based on the capability of hardware components and their configurations. In this paper, we define the throughput metric in the following way: "the number of packets successfully forwarded from switch i per unit of time".

$$Throughput = \frac{\sum_{p(i)} tranmsission\ success}{ţ} \quad (15)$$

(6). Rules Placement – Through flow rule installation, we avoid the packet loss, and service requirements meet the user/device QoS requirements. Also, rules placement must be reduced to avoid the flow table overloading.

$$Rule\ Placement = N(RP)\ at\ s(i)\ \in LC_i \quad (16)$$

Where RP refers to the number of rules placed in the switch i.

(7). Load Balance Rate – It is the rate of load balancing among switches and controllers. It is a positive metric so it requires a high value to obtain better QoS.

5.3. Comparative study

In this section, we demonstrate the experiment results for the proposed AI-aided SD-IoT model, as well as qualitative and quantitative comparison with the previous approaches in SDN and IoT. We have primarily focused on investigating the performance in four previous approaches, namely Sway [24], DNN-SR [38], FSM [39], and FRI [37]. In Table 7., we compare the performance by the aforementioned significant metrics.

5.3.1. End-to-end delay

The computation of delay for any kind of network is important to show the effectiveness of QoS. It is mainly caused by packets waiting in a queue. To avoid this delay, an optimum path is selected between the source to the destination. Fig. 9. indicates the performance of the end-to-end delay for the proposed model compared to the Sway and DNN-SDR.

From the analysis and trend line in Fig. 9., it is clear that the proposed model gains a lower end-to-end delay to transmit packets. In this paper, we differentiate the traffic arrived from various IoT devices. Based on the deep packet inspection, we classify the packets, and routing is implemented according to the classified traffic.

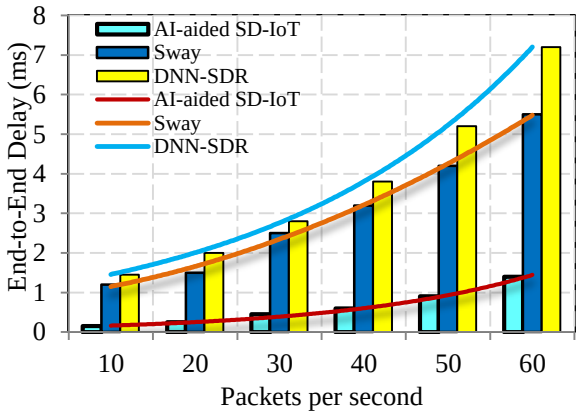


Fig. 9. End-to-end delay vs. Packets per second

In Sway, traffic type is considered, but it fails to select the optimum path, which results in high end-to-end delay. Similarly, DNN-SDR uses software-defined routers to send the packets to the destination router. It requires the optimum path and traffic differentiated route to forward the packets. The proposed work uses an optimum path according to the traffic type and decreases the end-to-end delay by above 45%.

Table 7. Qualitative comparison

Prior Works	Key Idea	Advantages	Limitations
FSM	Fractional level switch migration	(1). Does not require frequent flow migrations (2). Less overhead	(1). Not suited for a large-scale environment (2). Absolute switch prediction is not possible
Sway	Traffic aware QoS routing (delay-sensitive and loss-sensitive applications)	(1). The route is constructed for multimedia traffic (2). K-paths are determined	(1). Not suited for heterogeneous devices (2). Low throughput (3). Low scalability (4). Low Load balance rate
DNN-SDR	Load balanced routing via queue load rate prediction	(1). Network dimensionality is reduced (2). Achieved load balance rate	(1). Time-consuming (2). Poor scalability (3). Single point of failure
FRI	Flow rules placement in switches	(1). Predicts the best path for traffic (2). Low packet loss rate	(1). Low throughput (2). Traffic overhead is high (3). Large delay in best path selection
AI-aided SD-IoT	Traffic differentiation, QoS routing, rules placement and dynamic offloading	(1). High QoS (2). Less complexity (3). High scalability	-

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 8 (2020), pp. 1880-1895
 https://dx.doi.org/10.37624/IJERT/13.8.2020.1880-1895

5.3.2. Efficacy of packet loss rate

In a large-scale network, the packet drop rate is high in transmission and reception. The unit of packet loss rate is % and it must be low to show better performance. Fig. 10. shows the effectiveness of the packet loss rate of the proposed model compared to the previous approaches.

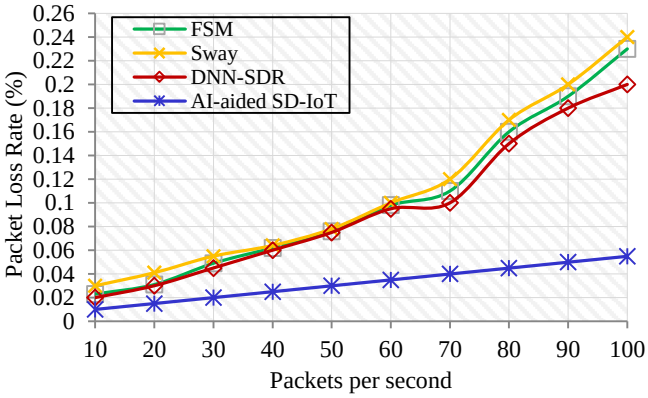


Fig. 10. Packet loss rate vs. Packets per second

When compare to Sway, DNN-SDR, and FSM, our proposed AI-aided SD-IoT model has shown good performance. This is achieved by optimum switches selection for transmitting packets and the selection of the method to provide the optimum solutions.

In SD-IoT, packet loss is a general issue, but it is not easy to recover it. In this paper, we effectively addressed this issue by proposing the optimum path and effective algorithms. In DNN-SDR, queue utilization is predicted by DNN for all SDRs, but the packet is transmitted via Dijkstra path selection. It causes higher packet loss rates. Our proposed model reduces the packet loss rate by up to 30%. Table 8. shows the packet loss rates comparison of the proposed model with FSM, DNN-SDR, and Sway.

Table 8. Packet loss rate

QoS works	# of packets sent	# of lost packets	Packet loss rate (%)
FSM	5000	226	4.51%
Sway	5000	175	3.24%
DNN-SNR	5000	125	2.5%
Proposed	5000	50	1%

5.3.3. Efficacy of switch failure rate

Providing load balancing in conditions where a large number of packets are processed from IoT devices is challenging. It must be taking into account that the size of the flow table is limited and that the switch may fail due to overfilling. The switch failure rate must be low because one of the basic assumptions of SDN is global network management, and this is reflected by minimizing the number of switch failures. Fig. 11. shows the switch failure rate versus packets per second. The computational delay between switch and controller is less because of partially connected topology created using the ISOMAP algorithm and the flow installed for some of the switches, which decreases the flow table overloading by the installation of new flow rules. Hence, the switch failure rate is decreased. Sway does not concentrate on the switch failure rate.

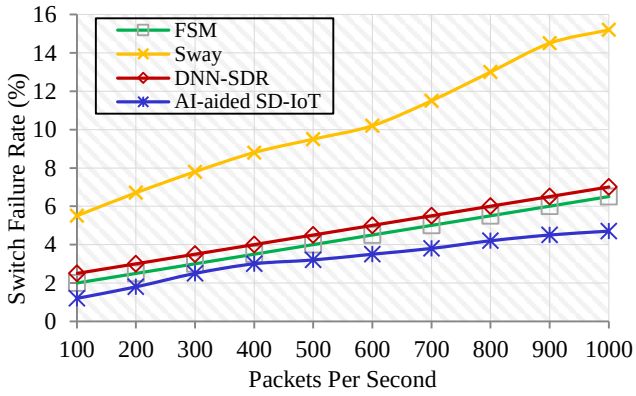


Fig. 11. Switch failure rate vs. Packets per second

When the packet is transmitted via frequent switches, it can cause failure even if the packet is forwarded on the same switch. To avoid this issue, switches failure must be handled promptly and it must be active when all packets are processed. In this paper, we considered multiple optimum criteria for switches routing. Due to the topology discovery and the type of network topology (fragmentation-optimized distributed controller), we achieved a lower switch failure rate. When compared to Sway, the AI-aided SD-IoT model has reduced the switch failure rate by up to 35%. Likewise, a 10.5% reduction was achieved compared to DNN-SDR and FSM.

5.3.4. Efficacy of controller failure rate

Similar to the switch failure, controller failures must be avoided in SDN since it directly affects the QoS. Controller failure happens when all the switches connected to the controller fail to process the incoming request.

Today, applications requested by the IoT users are different. Some of them are delay tolerant, and some of them are delay intolerant. For sensitive IoT applications, the delay must be lower. While handling a large number of applications with various QoS requirements, the computing resources for the controller may become unavailable. To overwhelm this problem, load balancing is introduced, but the effective load balancing mechanisms are still required. The performance of the controller failure rate for the proposed model and FSM, Sway, and DNN-SDR is given in Fig. 12.

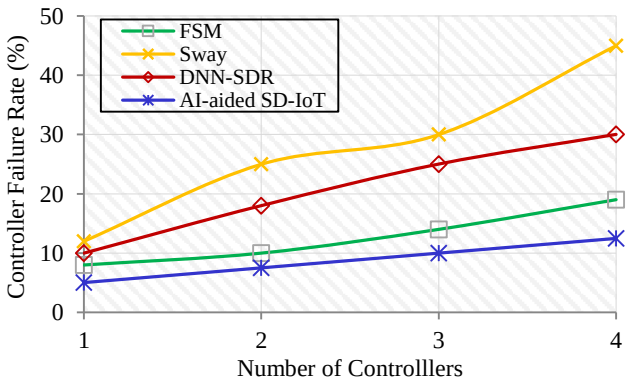


Fig. 12. Controller failure rate vs. Number of controllers

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 8 (2020), pp. 1880-1895
 https://dx.doi.org/10.37624/IJERT/13.8.2020.1880-1895

In comparison to the previous approaches, our model achieved a lower controller failure rate. The DC is responsible to take the action of offloading the packets in a dynamic time interval. It increases QoS efficiency and decreases the controller failure rate. When compared to Sway, FSM, and DNN-SDR, our proposed model has reduced controller failures by 15%.

5.3.5. Efficacy of throughput

It is a significant and positive metric in SD-IoT. It is reduced due to traffic congestion at large scale networks. In a dense network area, a huge number of packets are forwarded and received. It results in poor performance in terms of network throughput. Fig. 13. demonstrates the performance of the throughput versus network load.

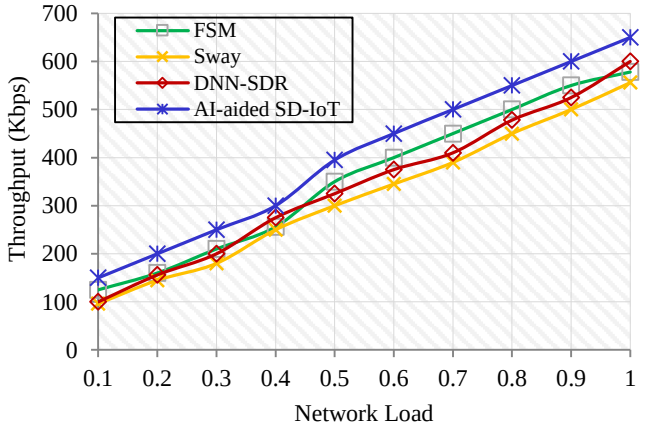


Fig. 13. Throughput vs. Network load

In sway, FSM, and DNN-SDR, the network traffic pattern is not highly concentrated and it is not based on the applications request from IoT devices. To meet this demand, in this paper, we introduce two core concepts, namely topology discovery and traffic differentiation by Isomap and deep packet inspection method. The proposed AI-aided SD-IoT model increases throughput by up to 12% compared to FSM, Sway, and DNN-SDR.

5.3.6. Efficacy of rules placement

In this paper, we first achieved the installation of flow rules in routing effectively. With the use of partially connected topology and all LCs in the network the switch's relationships can be verified.

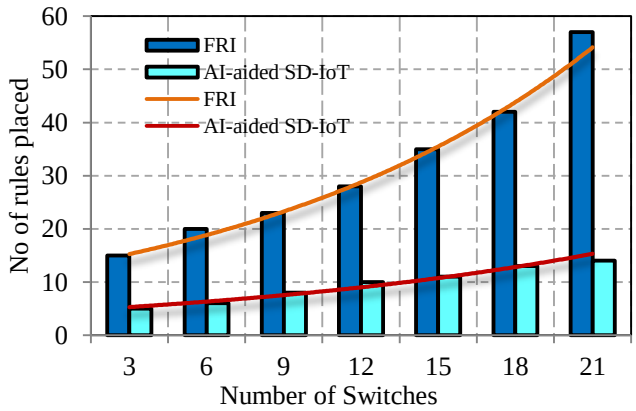


Fig. 14. Number of switches vs. No. of rules placed

A minimum degree of a switch in a controller is 3 and based on that the flow rules are deployed over the switch. This step helps to reduce the number of rules placement and prevents the flow table overloading. Fig. 14. shows the performance of the rule placement versus the number of switches.
We compared our proposed AI-aided SD-IoT model with FRI. In FRI, large numbers of flow rules are deployed, which increases the size of the flow table. When the switches are overloaded, packets migration is employed. A large number of switches migration degrade the QoS level and affect the controller performance.

5.3.7. Efficacy of load balance rate

The proposed model introduced the concept of dynamic offloading of IoT device requests (packets/tasks) in fragmentation-optimized distributed controllers.

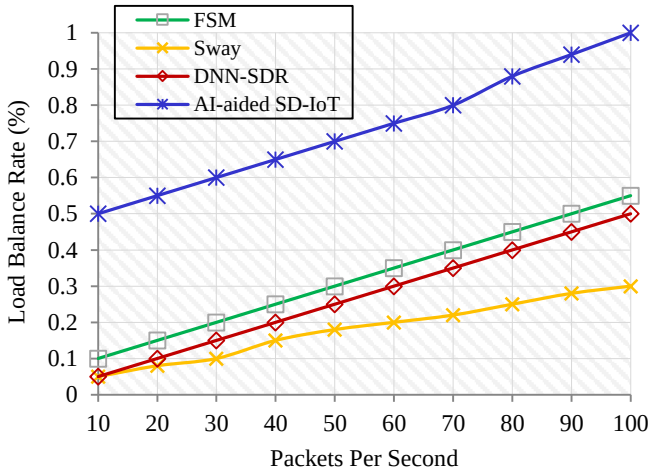


Fig. 15. Load balance rate vs. packets per second

The comparison of the load balance rate for the proposed and previous approaches is given in Fig. 15. To avoid the controller failures due to overloading issue, the multi-controller environment is established. The FSM approach only migrates the switches from the overloaded controller to the underloaded controller. It produces a better load-balancing rate on the control plane, but the frequent migration from one switch to another increases the overhead in the controller. Also, FSM does not concentrate on the data plane. In Sway and SDR, the load balance rate is reduced due to poor network management.

5.3.8. QoS validation

From the above analysis, we see that the proposed AI-aided SD-IoT model is capable of achieving high QoS in terms of end-to-end delay, packet loss rate, switch failure rate, controller failure rate, throughput, rules placement, and load balance rate. In particular, our proposed model achieves the best performance in QoS guaranteed factors, which is illustrated in Tables 9. and 10. We observe that the AI-aided SD-IoT model has satisfied QoS factors since it uses traffic differentiation, optimum routing via best switches, dynamic offloading, and rules placement.
In this paper, delay constraints, load balancing constraints, and flow constraints are satisfied. In traffic differentiation and

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 8 (2020), pp. 1880-1895
 https://dx.doi.org/10.37624/IJERT/13.8.2020.1880-1895

routing, the path is established based on the delay-sensitive and loss-sensitive applications flow.

Table 9. QoS guarantee factors

State-of-the-art	Delay	Through-put	Packet Loss Rate	Load Balance Rate	Scalability
Sway	✓	×	✓	×	×
DNN-SDR	✓	✓	✓	✓	×
FSM	✓	✓	✓	✓	×
FRI	✓	×	×	×	✓
AI-aided SD-IoT	✓	✓	✓	✓	✓

✓ - TRUE (Does meet the QoS requirement)
× -FALSE (Does not meet the QoS requirement)

Table 10. Quantitative comparison

QoS Parameters	FSM	Sway	DNN-SDR	FRI	AI-aided SD-IoT
End-to-End Delay (ms)	-	3.016	3.7416	-	0.625
Packet Loss Rate (%)	0.1029	0.1098	0.0955	-	0.0325
Switch Failure Rate (%)	4.25	10.27	4.75	-	5.2
Controller Failure Rate (%)	12.75	28	20.75	-	8.75
Throughput (Kbps)	357.9	321.2	344.4	-	404.5
Rules Placement	-	-	-	≅ 31	≅ 10
Load Balance Rate (%)	0.325	0.181	0.275	-	0.737

5.4. Motivating application

The usefulness of our proposed AI-aided SD-IoT model is given in this section by providing an example. We have tested our proposed model in Industry 4.0 applications, which is also referred to as IIoT. There are several IoT devices deployed in the industrial environment, namely RFID, BLE, Modbus, CANbus, EtherCAT, Profibus, and Profinet. From the IIoT devices, the task request is forwarded to the controller in XML or JSON format.

Fig. 16. shows the Industry 4.0 application implemented in SD-IoT. To ensure communication of IoT devices with Router, different communication protocols are used, such as HTTP, CoAP, MQTT, BLE, Modbus, and CAN-open Ethernet.

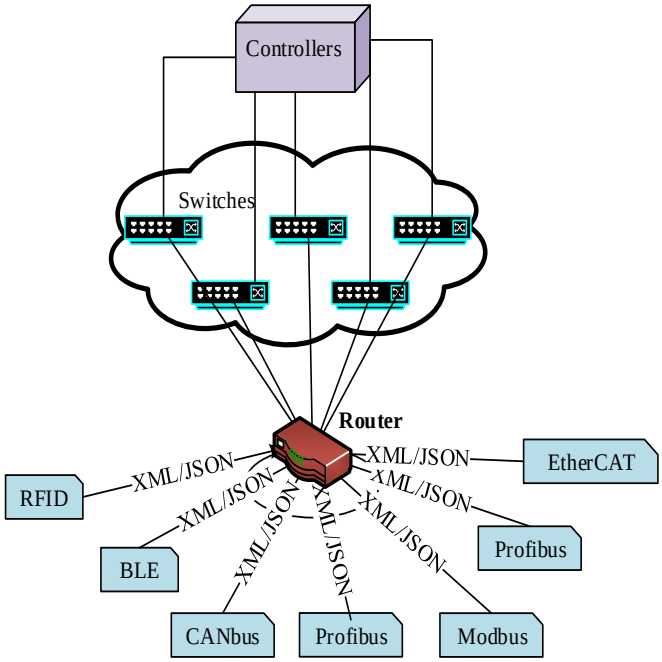


Fig. 16. Industry 4.0 in SD-IoT (Use case)

5.5. Computational complexity analysis

We evaluate the performance of the proposed solution by determining the amount of overhead and overload on the network. To analyze it, we calculated the computational complexity. IoT devices are resource-constrained and they must save energy to be able to operate for a longer time. Hence, we compute the time complexity of the proposed solution.

Time complexity refers to the order of growth the proposed algorithm requires in execution time for the given input. For representing the time complexity, Big-O notation is used. In this paper, the time complexity is predicted using three factors, namely, complexity, topology, and traffic pattern. Our proposed AI-aided SD-IoT model time complexity is compared with the previous state-of-the-art research and depicted in Tables 11. and 12.

Table 11. Time complexity computation

State-of-the-art	Complexity Type (Time)	Topology	Traffic Pattern
Sway	Non-deterministic polynomial time	×	×
DNN-SDR	Exponential time	✓	×
FSM	Log linear	×	×
FRI	Cubic	×	×
Proposed	Logarithmic	✓	✓

The statistical performance of time complexity is evaluated through simulations. Performance is compared with Sway, DNN-SDR, FSM, and FRI. *D* is the number of IoT devices and n represents the number of required operations. When network traffic is unevenly distributed to the SDN controller, then time complexity increases because switches are bandwidth constrained and need low response time for sensitive applications.

Table 12. Time complexity computation

Algorithm	Time Complexity (3 Cases)		
	Linear	Best	Worst
Sway	$\sim O(1)$	$\sim O(nD)$	$\sim O(n^2)$
DNN-SDR	$\sim O(nD)$	$\sim O(n^2D)$	$\sim O(2^ND)$
FSM	$\sim O(n^2D)$	$\sim O(n^2D)$	$\sim O(n^2D)$
FRI	$\sim O(nD)$	$\sim O(n^2D)$	$\sim O(n^2D)$
Proposed AI-aided SD-IoT	$\sim O(nD)$	$\sim O(4nD)$	$\sim O(n^2D)$

After the time complexity evaluation based on above factors, we have found that the proposed model requires $O(4n \times D)$, and $O(4n)$ represents execution time required for completing the four processes such as traffic differentiation, topology discovery, traffic differentiated routing, dynamic offloading and rule placement. From the time complexity analysis, if the number of devices increases and devices are resource-constrained, then the stronger mechanisms are required to improve the level of QoS.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel AI-aided SD-IoT model for traffic differentiated routing and dynamic offloading. We focused on AI concepts for all layers of the proposed model. To mitigate the issues of multi-controller environment-based load-balancing mechanisms, we presented fragmentation-optimized distributed controllers enabled SD-IoT environment.

We presented four new concepts in this paper to improve the QoS: traffic differentiation, topology discovery, traffic differentiated routing and rule placement, and dynamic offloading. We showed that the QoS is increased through the proposed concepts and algorithms, and we showed our proposed model is an effective and scalable solution for any kind of application environments (sensitive, non-sensitive, delay-tolerant, and delay intolerant). Moreover, we proved that our AI-aided SD-IoT model achieves 12% of reduced end-to-end delay, 30% of reduced packet loss rate, 35% of reduced switch failure rate, 15% of reduced controller failure rate, 12% of increased throughput, 65% of reduced rule placement and 30% of increased load balancing rate.

In the future, we plan to work on the 5G environment, which provides a high data rate and less latency in smart IoT applications, and we plan to test our proposed solution in the different real-world application scenarios.

REFERENCES

[1] Y. S. Yu and C. H. Ke, "Genetic algorithm-based routing method for enhanced video delivery over software defined networks," *Int. J. Commun. Syst.*, vol. 31, no. 1, pp. 1–13, 2018.

[2] A. K. Rangisetti, T. V. Pasca S., and B. R. Tamma, "QoS Aware load balance in software defined LTE networks," *Comput. Commun.*, vol. 97, pp. 52–71, 2017.

[3] H. Sufiev, Y. Haddad, L. Barenboim, and J. Soler, "Dynamic SDN controller load balancing," *Futur. Internet*, vol. 11, no. 3, pp. 1–21, 2019.

[4] S. Causevic and M. Begovic, "Optimizing Traffic Routing in Different Network Environments Using the Concept of Software-Defined Networks," in *Proceedings of the 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2019, pp. 409–414.

[5] Y. C. Chang, W. X. Cai, and J. W. Jhuang, "Bacteria-inspired communication mechanism based on software-defined network," *2018 27th Wirel. Opt. Commun. Conf. WOCC 2018*, pp. 1–3, 2018.

[6] M. Begović and H. Bajrić, "Solving Management Constraints of Traditional Networks using the Concept of Software Defined Networking," *Int. J. Soft Comput. Eng.*, vol. 7, no. 5, pp. 7–12, 2017.

[7] Y. Kyung and J. Park, "Prioritized admission control with load distribution over multiple controllers for scalable SDN-based mobile networks," *Wirel. Networks*, vol. 25, no. 6, pp. 2963–2976, 2019.

[8] J. Cui, Q. Lu, H. Zhong, M. Tian, and L. Liu, "A Load-Balancing Mechanism for Distributed SDN Control Plane Using Response Time," *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 4, pp. 1197–1206, 2018.

[9] Y. W. Ma, J. L. Chen, Y. H. Tsai, K. H. Cheng, and W. C. Hung, "Load-Balancing Multiple Controllers Mechanism for Software-Defined Networking," *Wirel. Pers. Commun.*, vol. 94, no. 4, pp. 3549–3574, 2017.

[10] T. Hu, P. Yi, J. Zhang, and J. Lan, "Reliable and load balance-aware multi-controller deployment in SDN," *China Commun.*, vol. 15, no. 11, pp. 184–198, 2018.

[11] C. Wang, B. Hu, S. Chen, D. Li, and B. Liu, "A Switch Migration-Based Decision-Making Scheme for Balancing Load in SDN," *IEEE Access*, vol. 5, no. c, pp. 4537–4544, 2017.

[12] T. Hu, P. Yi, J. Zhang, and J. Lan, "A distributed decision mechanism for controller load balancing based on switch migration in SDN," *China Commun.*, vol. 15, no. 10, pp. 129–142, 2018.

[13] H. Xu, H. Huang, S. Chen, G. Zhao, and L. Huang, "Achieving High Scalability Through Hybrid Switching in Software-Defined Networking," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 618–632, Feb. 2018.

[14] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "Fragmentation-based distributed control system for software-defined wireless sensor networks," *IEEE Trans. Ind. Informatics*, vol. 15, no. 2, pp. 901–910, 2019.

[15] J. Hua, L. Zhao, S. Zhang, Y. Liu, X. Ge, and S. Zhong, "Topology-Preserving Traffic Engineering for Hierarchical Multi-Domain SDN," *Comput. Networks*, vol. 140, pp. 62–77, Jul. 2018.

[16] F. Al-Tam and N. Correia, "On load balancing via switch migration in software-defined networking,"

IEEE Access, vol. 7, pp. 95998–96010, 2019.

[17] Y. Zhou, K. Zheng, W. Ni, and R. P. Liu, "Elastic Switch Migration for Control Plane Load Balancing in SDN," *IEEE Access*, vol. 6, no. c, pp. 3909–3919, 2018.

[18] K. L. Dias, M. A. Pongelupe, W. M. Caminhas, and L. de Errico, "An innovative approach for real-time network traffic classification," *Comput. Networks*, vol. 158, pp. 143–157, Jul. 2019.

[19] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Neural network architecture based on gradient boosting for IoT traffic prediction," *Futur. Gener. Comput. Syst.*, vol. 100, pp. 656–673, 2019.

[20] C. Yu, J. Lan, J. C. Xie, and Y. Hu, "QoS-aware traffic classification architecture using machine learning and deep packet inspection in SDNs," *Procedia Comput. Sci.*, vol. 131, pp. 1209–1216, 2018.

[21] A. Azzouni and G. Pujolle, "NeuTM: A neural network-based framework for traffic matrix prediction in SDN," *IEEE/IFIP Netw. Oper. Manag. Symp. Cogn. Manag. a Cyber World, NOMS 2018*, pp. 1–5, 2018.

[22] M. M. Tajiki, B. Akbari, M. Shojafar, and N. Mokari, "Joint QoS and congestion control based on traffic prediction in SDN," *Appl. Sci.*, vol. 7, no. 12, pp. 1–15, 2017.

[23] J. Park, J. Hwang, and K. Yeom, "NSAF: An Approach for Ensuring Application-Aware Routing Based on Network QoS of Applications in SDN," *Mob. Inf. Syst.*, vol. 2019, 2019.

[24] N. Saha, S. Bera, and S. Misra, "Sway: Traffic-Aware QoS Routing in Software-Defined IoT," *IEEE Transactions on Emerging Topics in Computing*, IEEE Computer Society, pp. 1–12, 2018.

[25] C. Lin, K. Wang, and G. Deng, "A QoS-aware routing in SDN hybrid networks," *Procedia Comput. Sci.*, vol. 110, pp. 242–249, 2017.

[26] Y. C. Wang and S. Y. You, "An Efficient Route Management Framework for Load Balance and Overhead Reduction in SDN-Based Data Center Networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 4, pp. 1422–1434, Dec. 2018.

[27] H. Wang, H. Xu, L. Huang, J. Wang, and X. Yang, "Load-balancing routing in software defined networks with multiple controllers," *Comput. Networks*, vol. 141, pp. 82–91, Aug. 2018.

[28] S. Misra and N. Saha, "Detour: Dynamic Task Offloading in Software-Defined Fog for IoT Applications," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1159–1166, May 2019.

[29] A. Akbar Neghabi, N. Jafari Navimipour, M. Hosseinzadeh, and A. Rezaee, "Nature-inspired meta-heuristic algorithms for solving the load balancing problem in the software-defined network," *International Journal of Communication Systems*, vol. 32, no. 4. John Wiley and Sons Ltd, 10-Mar-2019.

[30] S. Ejaz, Z. Iqbal, P. Azmat Shah, B. H. Bukhari, A. Ali, and F. Aadil, "Traffic Load Balancing Using Software Defined Networking (SDN) Controller as Virtualized Network Function," *IEEE Access*, vol. 7, pp. 46646–46658, 2019.

[31] Y. J. Chen, L. C. Wang, M. C. Chen, P. M. Huang, and P. J. Chung, "SDN-Enabled traffic-aware load balancing for M2M networks," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1797–1806, Jun. 2018.

[32] T. Hu, J. Lan, J. Zhang, and W. Zhao, "EASM: Efficiency-aware switch migration for balancing controller loads in software-defined networking," *Peer-to-Peer Netw. Appl.*, vol. 12, no. 2, pp. 452–464, Mar. 2019.

[33] H. Xue, K. T. Kim, and H. Y. Youn, "Dynamic load balancing of software-defined networking based on genetic-ant colony optimization," *Sensors (Switzerland)*, vol. 19, no. 2, Jan. 2019.

[34] L. Gupta, R. Jain, A. Erbad, and D. Bhamare, "The P-ART framework for placement of virtual network services in a multi-cloud environment," *Comput. Commun.*, vol. 139, no. October 2018, pp. 103–122, 2019.

[35] S. Bera, S. Misra, and A. Jamalipour, "FlowStat: Adaptive Flow-Rule Placement for Per-Flow Statistics in SDN," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 530–539, Mar. 2019.

[36] W. Li, Z. Qin, K. Li, H. Yin, and L. Ou, "A Novel Approach to Rule Placement in Software-Defined Networks Based on OPTree," *IEEE Access*, vol. 7, pp. 8689–8700, 2019.

[37] I. I. Awan, N. Shah, M. Imran, M. Shoaib, and N. Saeed, "An improved mechanism for flow rule installation in-band SDN," *J. Syst. Archit.*, vol. 96, pp. 1–19, Jun. 2019.

[38] H. Yao, X. Yuan, P. Zhang, J. Wang, J. Chunxiao, and M. Guizani, "Machine Learning Aided Load Balance Routing Scheme Considering Queue Utilization," *IEEE Trans. Veh. Technol.*, pp. 1–1, Jun. 2019.

[39] F. AL-Tam and N. Correia, "Fractional switch migration in multi-controller software-defined networking," *Comput. Networks*, vol. 157, pp. 1–10, Jul. 2019.

[40] X. Hou, M. Wu, and M. Zhao, "An Optimization Routing Algorithm Based on Segment Routing in Software-Defined Networks," *Sensors (Basel).*, vol. 19, no. 1, Dec. 2018.