SO-KDN: A Self-Organised Knowledge Defined Networks Architecture for Reliable Routing

S Gosh*

Department of Computer Science and Informatics, London South Bank University, UK

B EL BOUDANI

Department of Computer Science and Informatics, London South Bank University, UK

T DAGIUKLAS

Department of Computer Science and Informatics, London South Bank University, UK

I MUDDESAR

Department of Computer Science and Informatics, London South Bank University, UK

Abstract— “When you are destined for an important appoint-ment, you would obviously opt for the most reliable route instead of the shortest in order to be well prepared”. Modern networking is presently undergoing through a quantum leap. To cope up with ambitious demands and user expectations, it is becoming more complex both structurally and functionally. Software Defined Networking (SDN) happens to be an instance of such advancements. It has significantly leveraged the network programmability, abstraction, and automation. Eventually, with acceptance form all major network infrastructure such as 5G and Cloud, SDN is becoming the standard of future networking. Likewise, Machine Learning (ML) has become the trendiest skill-in-demand recently. With its superiority of analyzing data, makes it applicable for almost every possible domain. The attempt to applying the power of ML in networking has not been too long, it allows the network to be more intelligent and capable enough to take optimal decisions to address some of its native problems. This gives rise to Self- Organized Networking (SON). In this article, Routing using Deep Neural Network (DNN) on top of SDN is addressed. We proposed a Self-organized Knowledge Defined Network (SO-KDN) architecture and an intelligent routing algorithm, that reactively finds the most reliable route, i.e., a route having least probability of fluctuation. This reduces network overhead due to re-routing and optimizes traffic congestion. Experimental data show a mean 90% accurate forecast in reliability prediction.

CCS CONCEPTS • Networks ~ Network algorithms ~ Control path algorithms ~ Network control algorithms • Information systems ~ Information systems applications ~ Data mining ~ Data stream mining • Computing methodologies ~ Machine learning ~ Machine learning approaches ~ Neural networks

Additional Keywords and Phrases: SDN, SON, Deep Learning, Routing

ACM Reference Format:

Saptarshi Gosh, Brahim El Boudani, Tasos Dagiuklas and Muddesar Iqbal. 2021. SO-KDN: A Self-Organised Knowledge Defined Networks

Architecture for Reliable Routing. The 4th International Conference on Information Science and Systems ICISS 2021, March 17-19, 2021, Edinburgh, United Kingdom. ACM, New York, NY, USA, 11 pages. NOTE: This block will be automatically generated when manuscripts are processed after acceptance.

1 INTRODUCTION

IN the recent past, a paradigm shift in networking has been evident. With the introduction of Software Defined Net-working (SDN) [1] now we can think network programmability and automation in a whole new dimension. SDN decouples the control and data planes (CP & DP). DP runs on forwarding devices such as a switch (physical or virtual), and CP runs in the logically centralized server(s) that instruct(s) DP with appropriate forwarding rules using Southbound protocols such as OpenFlow [2]. Also, CP interacts with Application Plane (AP) using Northbound protocols such as REST which pro-grams the network in abstraction. The network controller node of any Infrastructure as a Service (IaaS) cloud platform uses SDN to provide communication among the compute instances. In practice, SDN implementation comes into two flavors, the first uses Bare-Metal switches running Linux such as Open V-Switches [2] (OVS) and the second way to virtualize the physical network infrastructure using SDN overlay such as VMWare NSX [3]. The 5G architecture white paper [4] describes the significance of self-organized networking (SON) including the three basic properties of it, Self-Configuration, Self-Optimization, and Self-healing. While the SDN provides abstraction and network programmability, another degree of self-reliance can be added with the use of a Machine Learning (ML) frameworks that learns from the network behavior. Mowei [5] has elaborately described the usage of ML in networking. Self-optimizing routes (Such as opportunistic routing [6]) plays a vital role in traffic management such as load balancing & congestion control on links. Learning the best routing policy and selecting the best path for packet transmission has been always the busiest research area in the field. The use of ML for finding the most reliable route is a novel area of research.

In this paper, an extended SDN architecture is proposed featuring ML framework on top of the application plane undergoing the following tasks:

1. Aggregating online monitoring information from a network, such as device & link utilization, change in route costs etc.
2. Learning a model from the gathered dataset and try to find patterns.
3. Using the model to predict the reliability of all the available routes to pick the most reliable one.

Furthermore, a routing algorithm named Most Reliable Route First (MRRF) is proposed that finds the most reliable route based on historical data to forward the future packets.

The rest of article is organized as follows, Section II dis-cussed the related work, Section III presents the architectural details, Section IV introduces the Algorithm, implementation details and experimental results are shown in section V and finally we conclude on section VI

1.1 Contributions

- A machine learning enabled SDN architecture is proposed with prototype implementation details.
- An Intelligent routing algorithm is proposed, which selects a most reliable route based on the history of change in cost.
- Experimental results confirm, with appropriately chosen parameters and techniques accuracy touches as good as 90% mark.

2 STATE OF THE ART

This article is an extension of our previous project named “Energy Aware Routing for SDN” [7], where we discussed how the processing load of forwarding devices affects the end-to-end bandwidth. We proposed a solution called Stochastic Temporal Edge Normalization (STEN), which considers the resource utilization of network devices and connecting links to find a path between a pair of nodes, that guarantees least delivery time. Our existing solution selects a path based on the total time taken for delivery. We realized this data is temporal and may contain fluctuation. As a result, the algorithm may choose a path when it offers very good value, and just after making the decision it goes to an inferior state. The algorithm then must trigger a re-route call, which enforces overhead and slows down the network. An ML approach may be useful here that observes the pattern of change in costs and can analytically predict the reliability of each route. Hence, instead of the least cost route, we prefer the most reliable route. Machine Learning provides solutions and methodologies for automating critical tasks like time series prediction and classification. According to Chollet [8],” search for useful representations of some input data, within a pre-defined space of possibilities, using guidance from a feedback signal. This simple idea allows for solving a remarkably broad range of intellectual tasks.”. From the perspective of traffic routing optimization, using Machine Learning in techniques to keep the performance and high availability of the network, is becoming more imperative than before.

In Boutaba et al [5]., the early use of intelligent traffic routing was Q-routing, a Reinforcement Learning (RL) algorithm, aiming accurate prediction of the best path and learn the optimum routing policies. It calculates the best path based on the transmission time over a link between a pair of nodes, where q-value refers to the amount of time a packet takes to reach its destination. This takes into consideration the time spent by a packet at certain node. This method depends on the length of the link over the size of the packet. However, the downside was that reliability was not taken into consideration since the path with the shortest link may not necessarily be the most reliable due to the stochastic nature of the network behavior. However, RL always assumes the given instances are Markovian which does not necessarily apply to traffic routing. Some of these instances cannot approximate using the length of the link and the time spent at each hop only. There are hidden factors to be explored e.g., the node cost. A major characteristic of RL and Q-learning specifically is that each input is processed independently and there is no link between each input or state [7]. Therefore, a system with memory capability to remember and learn behaviors of the previous inputs. A few years later, more researchers contributed to traffic routing optimization [9], [10]. Arroyo-Valles et al [11] have implemented Q-probabilistic routing. A node greedily chooses among its next-hop candidate neighbors the one that minimizes the cost of the route to the link. More recently, Pasca et al [10] have successfully implemented naive Bayes and C4.5 Decision Tree machine learning algorithms on multi-path packet forwarding in SDN. Also, Sinh et al [12] have used a distance- based algorithm called K-mean clustering to group traffics. This paper proposes the use of sliding window techniques and Deep Recurrent Neural Network to accurately choose and memories the most reliable path for traffic routing. This will lead to maximizing the links utilization and also selecting the packet for transmission. We are proposing a new concept for traffic routing via Deep Recurrent Neural Networks. To the best of our knowledge, this is the first time the concept is introduced in this area.

3 ARCHITECTURAL DESCRIPTION

Figure 1 depicts the detailed design on ML framework running on top of SDN, i.e. the schematic architecture of the test-bed. We extend the existing SDN architecture by appending a new layer on top of the application plane and termed as 'Analytics-Plane'. the analytics plane is responsible for running the machine learning algorithms to learn models from the data provided by the control plane. The complete data flow is indexed (in figure 1) and described below:

Step 1: OVS sends and receives packet to-and-fro the users' end devices. OpenFlow Controller instructs OVS with appropriate forwarding rules using OpenFlow (1.1). We have developed a custom push-agent based monitoring tool called ShellMon. The ShellMon-Clients run in the switches as an application and maintain a connection with ShellMon-Server, to periodically update monitoring data such as node and link utilization (1.2). In each period, various resource utilization information is collected and normalized using a mathematical model described in [6]. Normalized data are then stored in a relational database with a timestamp (2.2).

Step 2: network topology is fetched from the SDN controller using RESTFull API such as RESTConf (2.1). And the resource utilization data from all nodes are statistically aggregated (2.2).

Step 3: Node wise aggregated data are Summarized (3.2) and overlapped on top of the topology acquired pre-processed into a mathematical graph (3.1). As a result, a graph structure is formed where vertices and edges contain node & link utilization, respectively.

Step 4: Since the utilization info is updated periodically, it results in a time series. Also, between a pair of nodes, there may exist several routes. A route is an alternating sequence of the node and edges the cost of a route can be found by summing the cost of participating nodes and edges. Therefore, the cost of all the routes between all the pairs varies in time. The Analytics-plane use REST API to fetch that information and uses machine learning algorithms to perform tasks such as State Prediction of each network slice (4.1), optimized placement of VNFs within the slices (4.2) and analyze historical utilization data to find reliability of each route and selects the most reliable one (4.3). with respect to the context, (4.1 & 4.2) are beyond the scope of this article.

Step 5: Using RESTConf the most reliable route is fed back to the controller

Step 6: SDN Controller translates the route into switch wise OpenFlow entries and writes using OpenFlow protocol.

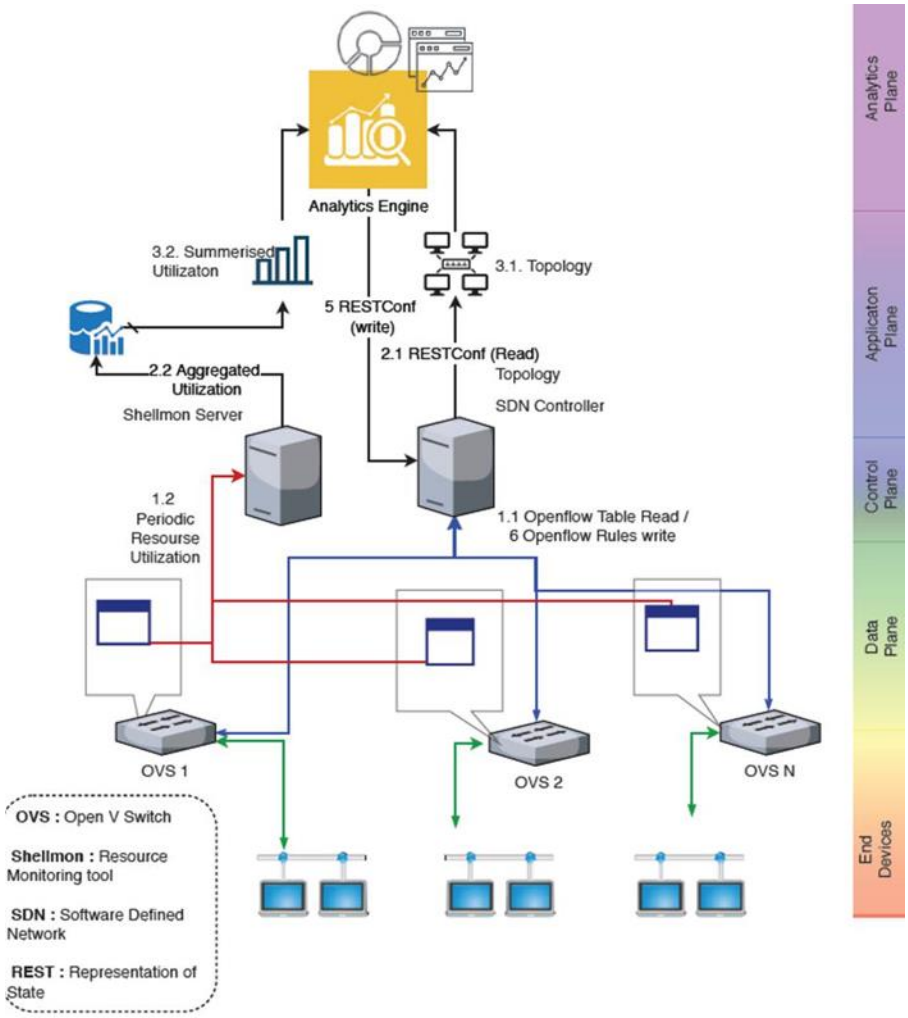

Fig. 1. Schematic Diagram depicting the functional building-blocks of proposed SO-KDN architecture. User interacts with End-devices, which connects the SDN. The Analytics Plane runs on top of SDN

4 THE MRRF ALGORITHM

The Most Reliable Route First (MRRF) algorithm (Figure 2) determines the most reliable path between a pair of nodes in a network topology. With the normalized edge by STEN [7] it gathers cost samples over a fixed time window, finds a pattern from the costs and calculates the reliability (e.g., a fluctuating link is less reliable than a stable one), and returns a path consists of most reliable links. The following subsection describes the steps (The implementation is publicly available at [13])

4.1 Getting the Initial Adjacency Matrix

The adjacency matrix (Adjn) is prepared by the application layer, with topology supplied by the SDN controller and costs supplied by the ShellMon-Server. The matrix represents the network in the following way, Diagonal

values represent node utilization Non-zero value at (i; j) location, represents the link cost between vertices i&j zero value at (i; j) location denotes non-adjacent pair of vertices.

4.2 Applying STEN on Initial Adjacency Matrix

Since the shortest path algorithm cannot work with self-loop, the node cost is normalized into the edge costs. Eventually, the edges get stretched as the weights of every edge gets added to the cost of its incident vertices. Let the normalized matrix is Adjs. After normalizing the matrix gets transformed as follows, All the diagonal values become zero All the non-zero non-diagonal values increase in such a way that the overall sum of matrix remain the same, to preserve the overall cost. All zero non-diagonal values remain same.

4.3 Calculating Quality & Reliability

For every edge, a finite number (d) of samples with normalized edge costs must be collected first. For every edge-costs, the median and the variance is calculated the quality (q) is expressed as a function of them, (in experiment, a weighted sum formula was used). Reliability is the Z-Transform over the set of all quality values of all the edges, which is performed to scale them into a close interval of [0; 1]. This is necessary for regularizing the dataset to leverage the learning algorithm.

4.4 Learning patterns from history

The Change in reliability over time is kept in a finite buffer (with a time-window), a machine learning algorithm finds patterns and forecasts reliability. Based on the prediction, the shortest path algorithm finds a route between a pair of vertices that tries to maximize the number of edges with high reliability. This fulfills the self- optimization property of SON. A route is determined as most reliable traffic flow start on it, this puts a load on the participating nodes and edges, eventually, it is quality falls. Therefore, a net routing request ignores it and finds an alternative route. Consequent the algorithm offers implicit load- balancing and meets the Self-healing property too. Self-configuration is offered by SDN, as it uses RESTConf to accept route information from northbound and translates it into OpenFlow entries into switches from southbound. As a result, all the participating switches gets configured automatically. Hence, we can be free to conclude that MRRF meets all the three criteria of SON.

4.5 Complexity Analysis

The algorithm first converts the network information into a mathematical graph structure. All the graph operations are performed by matrix transformation. Step A, B used matrix dot product, hence its O(1). Step C buffers d amount of data to start calculation hence O(d). Step D's complexity depends on what algorithm is used for learning. Since most of the learning algorithm are higher order, hence the complexity of the algorithm primarily depends on the complexity of the learning algorithm.

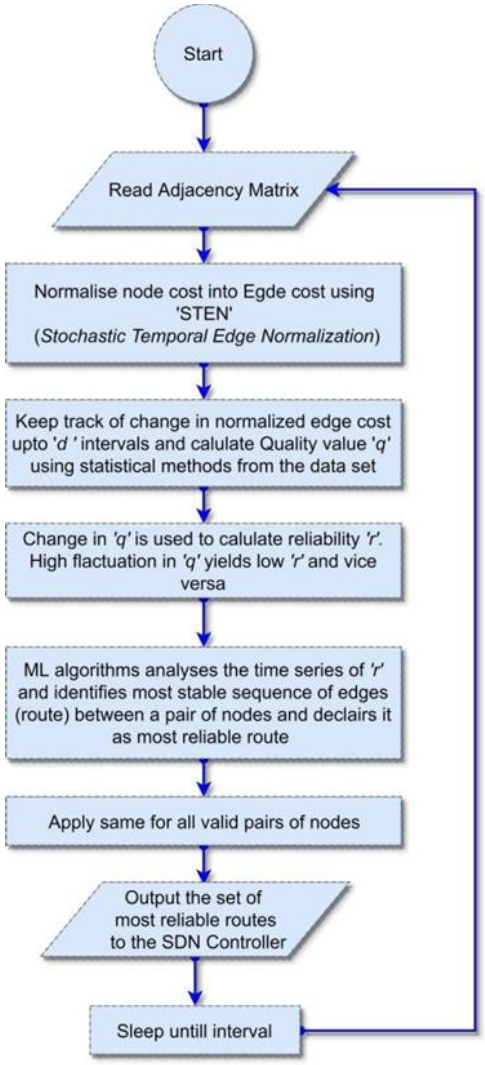

Fig. 2. Flowchart to describe steps of MRRF algorithm. It invokes STEN [7], as a part of execution to normalize the node cost into links, before calculating the reliability

5 ARCHITECTURE OF THE ANALYTICS-PLANE

In this section, the analytics plane's architecture is dis-cussed. It can leverage both network links and nodes utilization, using data collectors to harvest vast amounts of resources utilization data to help the networks to become self-driven. The figure 3 shows a detailed architecture of the Analytics-plane and its interaction with other network planes. It interfaces with the application plane of the SDN, to fetch ag-aggregated data, collected by controller and other Applications. Learning algorithm builds a model out of the dataset, which is eventually used to make various decision using prediction and reliability calculation. The decisions are fed back to the application layer to convey the control-plane for Self-Configuration.

5.1 Analytics Plane Framework Components

Working principals and data-flow model is described below:

1) **ShellMon Server** (Master Collector): Establishes and maintains a reliable & secure (TCP, SSH) connection with the client agents on the Data-plane. Aggregates the fetched data and store centrally for analysis.

2) **ShellMon Client** (Client Collector): Runs as agents within the Network devices. Collects, local utilization information and sends to the server periodically. Details of resources and their attributes are shown in Table 1.

3) **Data Aggregation API** (Restful Data Aggregator): To learn from the Collected data, it has to be aggregated and structured. A RESTful API is developed to fuse the controller and shellmon server information, that includes network topology, node and link utilization, respectively. The aggregated data is then accessed by the Analytics Plane to build its model.

4) **Resources Visualizer** (ResViz): A web-based data visualization / Dashboard tool, developed to have a global view of the network. ResVis makes use of the aggregator API, to fetch and project node-wise status (such that utilization, topology etc.). It uses Python Flask framework using Light-HTTP server, D3 library for visualization and runs on TCP port 6161.

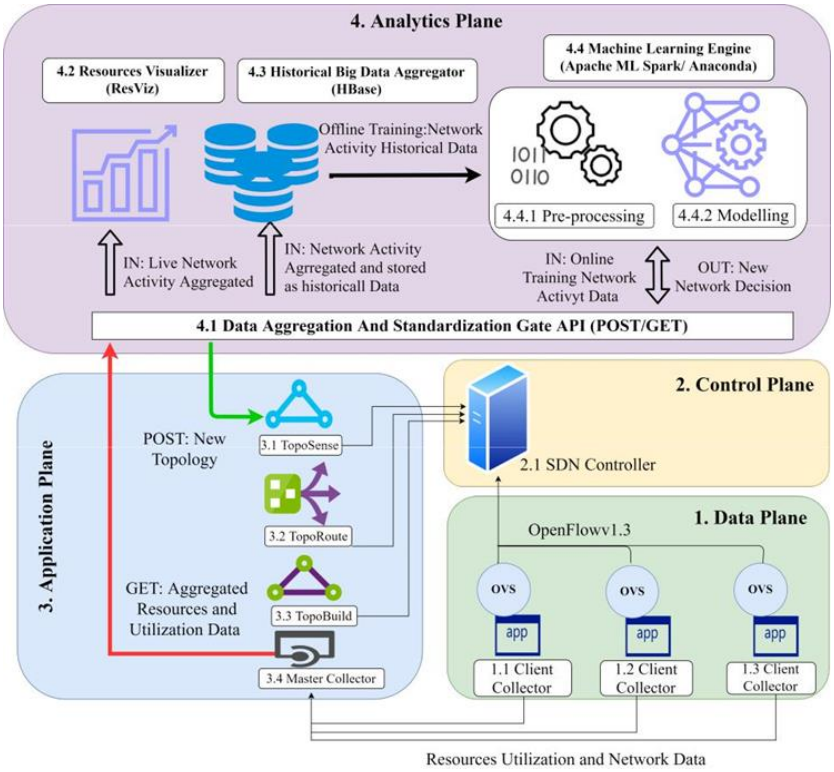

Fig. 3. Functional architecture of the Analytics Plane, along with its communication with Data, Control, Application planes.

5) Historical Big Data Aggregator (HBDA): To perform offline training and prediction by Deep Learning engine, a data repository is crucial. utilization data collected from the Aggregator API is processed into a individual time-series and stored centrally. A Cloudera-HBase server is used for this purpose.

6) Machine Learning Engine: This component takes care of all data pre-processing, offline and online training. It returns a trained model initially as an outcome of offline training however the model gets updates during online training whenever the trend changes. functionalists can be divided into four main unit:

- **Pre-processing:** Acts as a staging area for the model the training units. performs data acquisition, data quality checks and validations, imputing and standardization. Typically, 70% of the overall process time is spent on this phase.
- **Offline Training:** After the pre-processing tasks, the offline training starts by dividing the data into train-ing, validation and testing for the machine learning model. It utilizes the historical data from the repos-itory to train the model, predicts the networking characteristics to produce a decision such as VNF placement and state prediction.
- **Online Training:** It is used when the data is generated in a form of a sequence (such as time series). Network resource utilization is a form of a time series. The Topology is represented as a matrix, each element of the matrix represents a normalized link cost between a pair of nodes. Over the time a sequence of such matrices are received, making it a n n t tensor. where n be the number of nodes and t be the time.
- **Modelling:** The learning algorithm learns from the fed dataset and generates a model for prediction. As the subjected problem can be classified as a time series prediction type, RNN is chosen the base architecture.

Table I Resources utilization (of CPU, Memory and Network) gathered using shellmon client with their corresponding attributes and units of measure. Network attributes, operating frequency, Tx power, signal level and link quality are only available if the interface is of wireless type.

Resource	**Components & Units**							
CPU	Attributes	Physical Core Count	Virtual Core Count	Mean Frequency	Mean Load			
	Units	Number	Number	GHz	[0,1]			
Memory	Attributes	Volume	Mean Utilization	Mean Frequency				
	Units	GB	[0,1]	GHz				
Network	Attributes	Ingress Utilization	Egress Utilization	Link Speed	Operating Frequency	Tx Power	Signal Level	Link Quality
	Units	[0,1]	[0,1]	[Mbps]	[2.4 / 5 GHz]	mW	dbm	[0,1]

The learning process uses the Sliding-Window principal that only consider historical data up to a relevant timeframe. From the time series of varying network cost, it learns traffic pattern of each link, which yields their reliability. While Finding a shortest path, MRRF utilizes the link-reliability as weights and selects a path comprises of most reliable links. Hence the Self-Optimization criterion is met.

6 IMPLEMENTATION AND RESULTS

In this section the implementation phases of MRRF algorithm over the SO-KDN architecture is discussed. Results are given to validate the proposed Deep Neural Network architecture, that includes Number of Neurons, Type of algorithms and optimizer etc. However experimental details and step-by-step implementation guide including Apps mentioned in the paper, are made publicly available on GitHub [13].

6.1 Building the Network Architecture

Network traffic is generated using Ostinato (an Open-source traffic generator) over a Emulated SDN architecture in GNS3 with Open V-Switches (OVS) and Open Daylight (ODL) Controller. For Non-SDN setup, Cisco 7200 & Quagga soft-ware routers are used, with a Custom Overlay network using Python-NAPALM library.

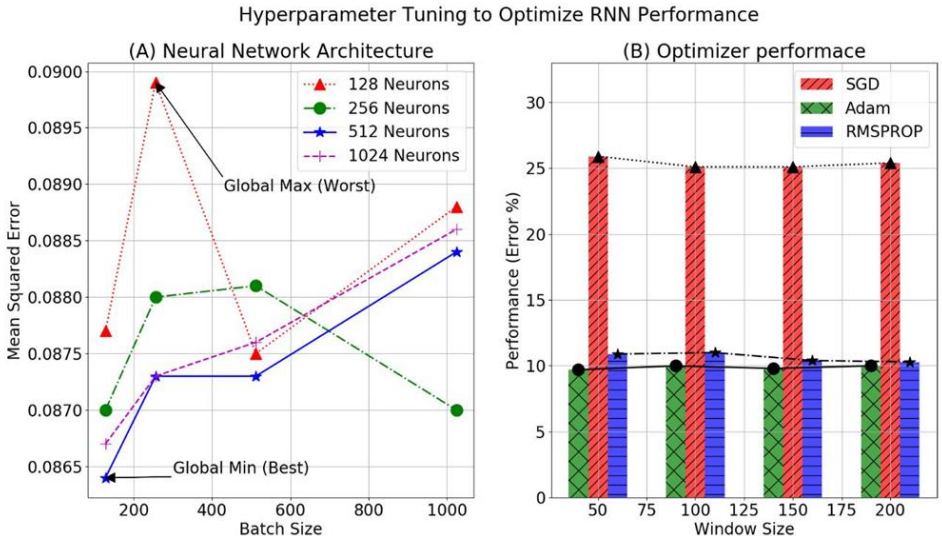

Fig. 4. (A) Comparison of accuracy (by mean squared error) with four network setups (128, 256,512 1024), the Global optima is reached with 128 Neuron at a batch size of 512. (B) compares three optimizer algorithms (SGD, Adam RMSPROP), over a varying window size of [20 200], on which Adam gives best result on average

6.2 Traffic Prediction and Reliability Analysis

In this section, the design of the Machine learning architecture is presented. We also introduce a few techniques used like hyper-parameters fine-tuning and choosing the best optimization algorithm.

1) hyper-Parameter Tuning: Recurrent Neural Network (RNN) is the de-facto choice of any time-series prediction problem in deep learning, hence is chosen as a type of learning model. hyper-parameter such as batch size and number of neurons are tuned from experimental data. Figure 4(A) depicts testing Mean Squared Error (MSE) cross-validation for 3 layers on a Deep Recurrent Neural Network using 10 epochs. The reason for this was to choose the appropriate number of neurons and the batch size for the training and validation datasets. This was measured using MSE Performance achieve the minimum error rate. As highlighted in bold, the optimum hyper-parameters were 128 neurons and 512 batch size at 0.08 MSE.

2) Optimization Algorithm: The figure 4(B) shows a comparison of the various optimizers proposed by [14]. For the LSTM model different sets of window sizes are tested. three main variants Gradient [14] Descent (SGD, ADAM & RMSPROP) is compared. as a proof of concept, results depict that predicting with a 200ms window size using Adam can achieve a mean error rate of 10%.

3) Scoring: The proposed technique performs traffic pre-diction on the normalized reliability of the links. Results confirm. With the appropriate hyper-parameters, reliability can be forecasted with a mean 90% accuracy. Therefore, MRRF claims in choosing the most reliable route to forward packets with a very high degree of precision.

7 CONCLUSION AND FUTURE SCOPE

In this paper the Implementation of SO-kDN architecture with a use case of intelligent routing algorithm (MRRF) using Deep learning framework (RNN, LSTM, Adam) is introduced. it learns from the network traffic behavior from an SDN platform and find a most reliable route rather than just shortest fulfilling the 5G-SON criteria. Experimental results confirm a high degree of accuracy. In the future, we are looking at improving the proposed model by automatically find the optimal window size for making it more self-reliant.

ACKNOWLEDGEMENT

This project is funded by MSCA-RISE - Marie Sklodowska-Curie Research and Innovation Staff Exchange (RISE) scheme under the project SONNET

References

[1] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koupavlou, “Software-Defined Networking (SDN): Layers and Architecture Terminology,” RFC 7426, Jan. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7426.txt

[2] OpenFlow Switch Specification, 1st ed., Open Networking Foundation, Jun. 2012.

[3] Vmware nsx data center for vsphere documentation. VMware. [Online]. Available: https://docs.vmware.com/en/VMware-NSX-Data-Center-for-vSphere/index.html

[4] G. A. W. Group. (2017, Dec.) View on 5g architecture. 5GPPP. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2018/01/5G-PPP-5G-Architecture-White-Paper-Jan-2018-v2.0.pdf

[5] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, “A comprehensive survey on machine learning for networking: evolution, applications and research opportunities,” Journal of Internet Services and Applications, vol. 9, no. 1, Jun 2018

[6] N. Chakchouk, “A survey on opportunistic routing in wireless communication networks,” IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2214–2241, 2015.

[7] S. Ghosh, T. Dagiuklas, and M. Iqbal, “Energy-aware IP routing over SDN,” in 2018 IEEE Global Communications Conference (GLOBECOM). IEEE, dec 2018.

[8] . F. Chollet. Deep learning with python. [Online]. Available: https://github.com/fchollet/deep-learning-with-python-notebooks

[9] T. V. P. S, S. S. Prasad, and K. Kataoka, “Ampf: Application-aware multipath packet forwarding using machine learning and sdn.”

[10] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, “Machine learning in wireless sensor networks: Algorithms, strategies, and applications,” IEEE Communications Surveys & Tutorials, vol. 16, no. 4, pp. 1996– 2018, 2014.

[11] R. Arroyo-Valles, R. Alaiz-Rodriguez, A. Guerrero-Curieses, and J. Cid-Sueiro, “Q-probabilistic routing in wireless sensor networks,” in 2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information. IEEE, 2007.

[12] L.-V. Le, D. Sinh, B.-S. P. Lin, and L.-P. Tung, “Applying big data, machine learning, and SDN/NFV to 5g traffic clustering, forecasting, and management,” in 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft). IEEE, Jun 2018.

[13] S. Ghosh. Self-organised knowledge defined network architecture. London South Bank University. [Online]. Available: https://github.com/rishiCSE17/SO-KDN

[14] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, “Machine learning for networking: Workflow, advances and opportunities,” IEEE Network, vol. 32, no. 2, pp. 92–99, mar 2018.