TO ENHANCE MANAGEMENT OF SOFTWARE DEFINED NETWORKING USING DEEP LEARNING METHOD

Szu-Yu Liu, Institute of Information Management, National Chiao Tung University, Hsinchu City, Taiwan, ROC, 886-3-5712121#57421, candice102403006@gmail.com
Cheng-Yuan Ku, Institute of Information Management, National Chiao Tung University, Hsinchu City, Taiwan, ROC, 886-3-5712121#57413, cooper.c.y.ku@gmail.com
David C. Yen, Jesse H. Jones School of Business, Texas Southern University, Houston, TX 77004, USA, 713-313-7215, david.yen@tsu.edu

ABSTRACT

Network administrators must have accurate networking information to provide good management. This requirement was not feasible in early days; however, emerging Software-Defined Networking (SDN) has made it possible nowadays. The controller periodically monitors utilization of each link and the potential overused links could be identified in advance. For that reason, removing these bottleneck links from future routing paths seems to be a rational choice. In this paper, we propose an enhanced scheme with dynamic thresholds based on Long Short-term Memory (LSTM). The simulation results indicate that it can improve network performance and enhance quality of service (QoS) as well.

Keywords: Software-Defined Networking, Management and Traffic Engineering, Congestion Control, Quality of Service, Long Short-term Memory

INTRODUCTION

The network has played an increasingly important role in our daily lives [3,4]. As traffic on many networks increases, congestion continues to occur and leads to unacceptable network performance. In the past, telecommunication operators increase network capacity by using high performance switches and routers to improve networking efficiency for a long time. However, this over-provisioning solution may have a low utilization rate and sometimes cannot effectively solve the bottleneck problem. This is because of the inefficiency of the existing routing architecture. The main selected routes are frequently used so that it eventually leads to overuse and congestion of these links and leaves other links idle and wasted. Multiprotocol Label Switching (MPLS), as a solution for traffic engineering, decides the route between the source and destination and clearly controls the traffic distribution [1]. However, the labels in MPLS are statically configured and have no real-time capability to reconfigure. Hence, optimal arrangement of routing loads in real-time becomes quite necessary, especially for high-speed networks. On the other hand, the rapid growth of network traffic also emphasizes the importance of network monitoring, management, planning and traffic engineering. A lot of monitoring technologies in traditional networks require the separate hardware mechanism or software configuration that makes it a really tedious and expensive task. Therefore, the emergence of SDN technology apparently solves this critical issue.

Within the SDN architecture, the control plane and data plane are separated; therefore, it is able to provide network administrators the convenient tools for traffic monitoring and some easy ways to implement new control policies [11]. Park et al. propose an SDN TE solution to provide the failure recovery mechanism for network components and then load balance comes in nature reactively or actively [9]. Mohan et al. think that the active method can speed up the failure recovery; however, the link nodes do not interact with the backup path but using the pre-computed rules [8]. In the above-mentioned methods, the choice

of the alternate paths or the backup path does not take into account the current network utilization and traffic statistics. Sminesh et al. propose a scheme that models a Bayesian Network using the observed port utilization and residual bandwidth and then decides whether the new alternate path can handle the flow load in SDN [12]. Their method somewhat improves network performance of SDN. However, the thresholds of their solution are fixed. If the threshold of each link can be dynamically decided based on networking situation, we believe that it may tremendously enhance the network performance. Thus we propose embedding a deep-learning algorithm LSTM in the routing procedure to select the optimal and dynamic thresholds using real-time data collected in SDN controller. Simulation results show that the proposal indeed fulfills our speculation.

RELATED WORKS

Software Defined Networking

Software Defined Networking is an innovative network architecture proposed by scholars at Stanford University and has received widespread attention in recent years. The definition of SDN provided by the Open Network Foundation (ONF) is the most acceptable one [2]. According to the ONF, SDN is an architecture that is dynamic, manageable, adaptable, and cost-effective. The control plane is physically separate from the data plane and one control plane controls several data plane devices. The conceptual SDN architecture is shown in Figure 1 [2]. Network intelligence is logically concentrated in the SDN controller that maintains a global observation of the network. This centralized and real-time view makes the controller ideal to realize functions of network management [7].

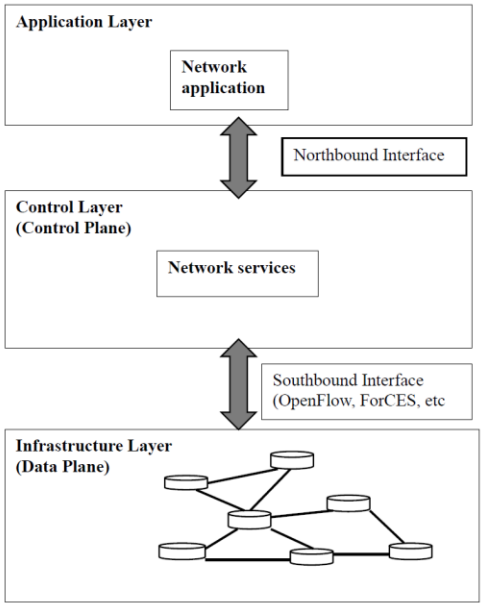

Figure 1 SDN architecture [2]

Traditional networking is mainly implemented in dedicated devices and hardware such as switches, routers and controllers. It is a static and inflexible structure. Compared with the traditional network architecture, SDN has the following distinguishing features [2].

- Centralized control: the SDN controller stores the information and states of the whole network including its topology, real-time changes of network statuses, and global application requirements [5].

- Programmability: network operators can dynamically program the devices of data forwarding layer so as to optimize the allocation of resources.
- Open standards: the forwarding device has a unified interface for communication with controller in SDN. It has nothing to do with any device vendors. Therefore, SDN controller can conveniently acquire data of network statuses to schedule networking traffic.

Long Short-term Memory

LSTM is a kind of recurrent neural network (RNN) with the capability of remembering values from early stages for the purpose of future use [13]. The LSTM algorithm is really suitable for classification, processing and prediction based on time series data, as there may be a lag of unknown duration between important events in the time series. The main feature of LSTM is its gate mechanisms: input gate, forget gate and output gate [14]. The definitions of these gates is provided in Table 1 as below.

Table 1 LSTM gate definition

LSTM gates	Definition
Input gate	How many new state vectors are added to the subsequent operations.
Forget gate	Filter the previously calculated status values to select which information to forget.
Output gate	How much internal state information is exposed to the outer neural network (higher layer neural network or next time step).

These three gates perform the same operation and the values of the vectors are compressed to 0 or 1 by the sigmoid function. 0 means discarding the input value. 1 means allowing this input value pass through the gate and participating in the subsequent operations. Then it is multiplied by the vector as a state value element by element to decide whether have the input vector update the state value. The LSTM module is shown in Figure 2 as follows.

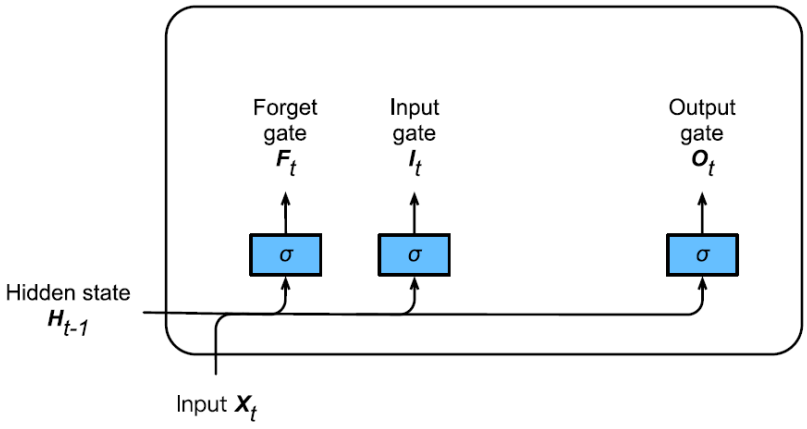

Figure 2 Input, forget, and output gates in an LSTM [14]

PROPOSED FRAMEWORK

As introduced earlier, Sminesh et al. propose a scheme to observe port utilization and residual bandwidth and then decide whether the new alternate path can handle the flow load in SDN [12]. Thus, in this paper, we propose a revised version to enhance routing performance using LSTM algorithm so as to make

dynamic decisions. The process of flow path settlement is described as in Figure 3. LSTM is used on the right-hand side to decide the optimal and dynamic thresholds.

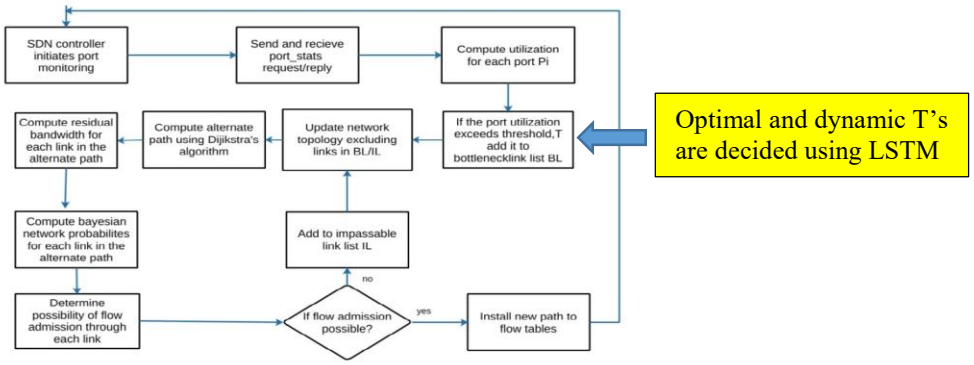

Figure 3 Flow diagram of proposed method [12]

The algorithm of LSTM used in the above flow diagram is described in the following Figure 4.

Algorithm: Pseudocode of LSTM algorithm

```
Procedure LSTM(train, test, epoch, neurons)
    x ← train
    y ← test
    model = Sequential()
    model.add(LSTM(neurons), return_sequences=True)
    model.compile(loss='categorical_crossentropy',
                  optimizer='adam')
    for each i in range do
       model.fit(x, y, epochs=50)
    end for
    model.summary()
    return model
End Procedure
```

Figure 4 Pseudocode of LSTM algorithm

PERFORMANCE EVALUATION

Simulation Tools

For simulating networking environment and proposed method, we choose some open sources to do so. The tools used in our experiment include Mininet, Ryu, Distributed Internet Traffic Generator (D-ITG), Iperf and python library Keras. Furthermore, we select the scheme of Sminesh et al. [12] as the comparison baseline. The following Table 2 shows the information regarding the simulation hardware.

Table 2 The hardware specifications

Device name	Description
CPU	Intel(R)Core(TM) i5-2500 CPU @3.30GHz
RAM	8GB
OS	Ubuntu 16.04.1 LTS

Mininet is a system that allows prototyping SDN networks and tests network performance on a single computer rapidly. It contains simple command line tools and API. Mininet can create a virtual network by

placing host processes in the network namespace and connect them to a virtual veth pair. The host is emulated as a bash process running in the network namespace so that any code can normally be run on a Linux server. The switches in the Mininet are software-based switches such as OpenvSwitches or OpenFlow reference switches. The links are virtual Ethernet pairs that exist in the Linux kernel and connect the emulated switches to the emulated host.
The Ryu controller provides software components with well-defined APIs that make it easy for developers to create new network management and control applications. Ryu controller is an open SDN controller designed to increase network flexibility.
Keras is an advanced neural networks API coded in Python that runs as a backend with TensorFlow, CNTK, or Theano. The development of Keras mainly focuses on supporting the rapid prototyping.
D-ITG is a platform that is capable to produce accurate compliance traffic (IPv4 and IPv6 traffic) defined by the departure time between packets (IDT) and packet size (PS). It is also a network measurement tool that measures most of common performance parameters such as throughput, delay, and packet loss.
Iperf is an open source software coded in C and run on a variety of platforms including Linux, Unix and Windows. It is a performance measurement tool for TCP/IP and UDP/IP. It can test the maximum bandwidth of TCP by tuning various parameters and report statistics such as bandwidth, delay, maximum segment and maximum transmission unit size.

Networking topology

The simulation topology we select is fat tree as shown in Figure 5. Fat tree is a scalable universal network class topology [6] and is based on the complete binary tree. Fat tree has identical bandwidth at any bisections and each layer has the same aggregated bandwidth. We consider fat tree topology with 4 pods to implement the traffic scheduling method and try to optimize the network situation.

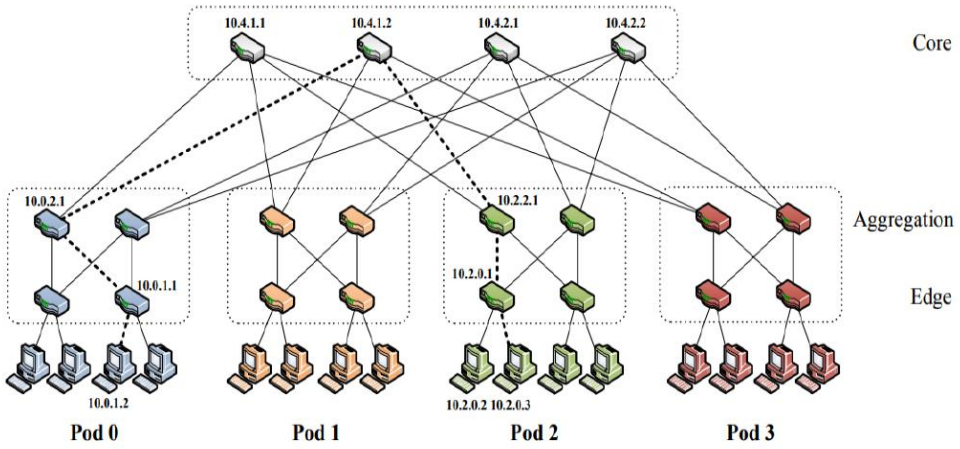

Figure 5 Simulation topology

Two Strategies and Performance Parameters

The proposed scheme is evaluated in two strategies that change the number of situation classification. The packet sizes we use are set between 1000 and 10000 bytes. The performance of SDN is analyzed in terms of packet loss, throughput and average end-to-end delay [10].
The definition of the packet loss is the number of packets that cannot reach the destination. It is expressed as the percentage of loss based on the following equation (1).

$$\text{Packet Loss} = \frac{Total\ number\ of\ packets\ dropped}{Total\ number\ of\ packets\ send} * 100\% \quad (1)$$

The definition of throughput is the amount of data received in Mbps per unit time based on the following equation (2).

$$\text{Throughput} = \frac{Data\ bits\ received\ in\ flows}{Total\ time} \quad (2)$$

The average end-to-end delay is the time it takes for all the packets to reach their destination averaged across all the transmitted packets based on the following equation (3).

$$\text{Average end-to-end delay} = \frac{\text{End-to-end delay of each flow in total}}{Total\ number\ of\ flows} \quad (3)$$

Strategy 1 Performance Analyses

The network topology links are configured as 10Mbps bandwidth. For the first strategy, LSTM algorithm classifies network situation into three categories and then determines the optimal threshold adjustment based on these three categories. The D-ITG packet generator is used to send 1000, 2000, 3000, 4000 and 5000 byte packets from Pod1 to Pod2 by varying the number of packets transmitted per second. Other traffics are set to be very bursty. The performance of the proposed system (experimental group) is analyzed with packet loss percentage, throughput and end-to-end delay and the results are compared with the baseline method (control group). The percentages of packet loss with different packet rates are listed in Table 3 and demonstrated in Figure 6.

Table 3 Packet losses in percentage with different packet rates (strategy 1)

Number of packets sent (pps)	Control group (packet loss %)	Experimental group (packet loss %)
1000	26.89	19.63
2000	27.12	20.19
3000	27.66	21.75
4000	27.59	22.02
5000	27.83	22.15

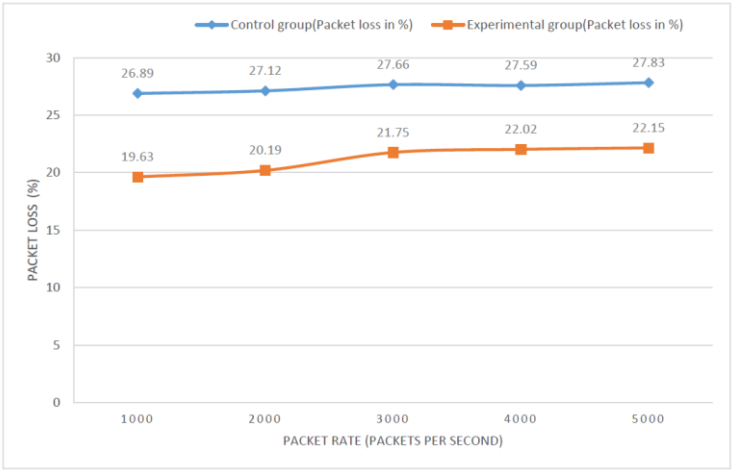

Figure 6 Packet losses with different packet rates (strategy 1)

It is observed that the proposed framework provides lesser packet losses than the compared framework. Our scheme considerably reduces the number of packets dropped.
Throughputs of two frameworks observed in this simulation with different packet rates are shown in Table 4.

Table 4 Throughputs in Mbps with different packet rates (strategy 1)

Number of packets send (pps)	Control group (Mbps)	Experimental group (Mbps)
1000	6.820	7.568
2000	6.831	7.454
3000	6.756	7.333
4000	6.766	7.307
5000	6.733	7.245

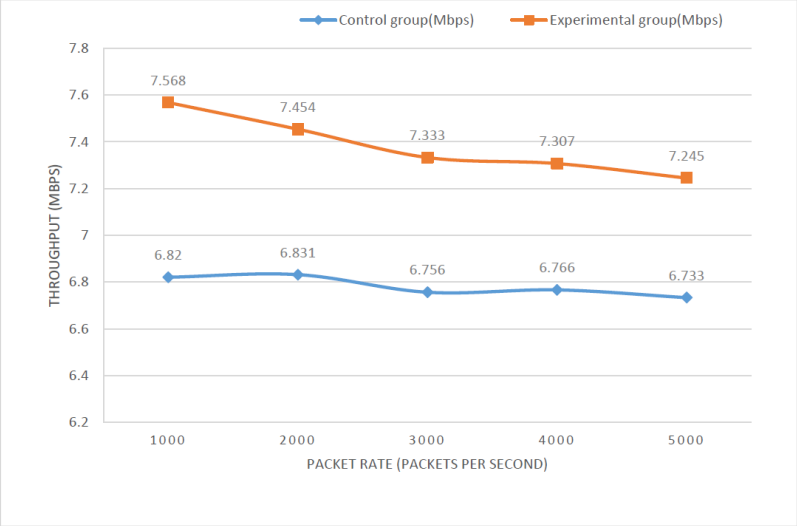

Figure 7 Throughputs with different packet rates (strategy 1)

It is observed that throughputs of the proposed framework are better than the compared framework. By using the LSTM algorithm to learn better thresholds before rerouting, the proposed framework reroutes packets through the alternate path without bottleneck links dynamically. Hence it further reduces packet losses and achieves high throughput even up to 7.568 Mbps. The throughputs with varying packet rate are depicted in Figure 7.
The average end-to-end delays of two frameworks with varying packet rates in this simulation are listed in Table 5.

Table 5 Average end-to-end delays in second with different packet rates (strategy 1)

Number of packets send (pps)	Control group (s)	Experimental group (s)
1000	1.090	1.096
2000	1.098	1.092
3000	1.093	1.091
4000	1.092	1.089
5000	1.089	1.093

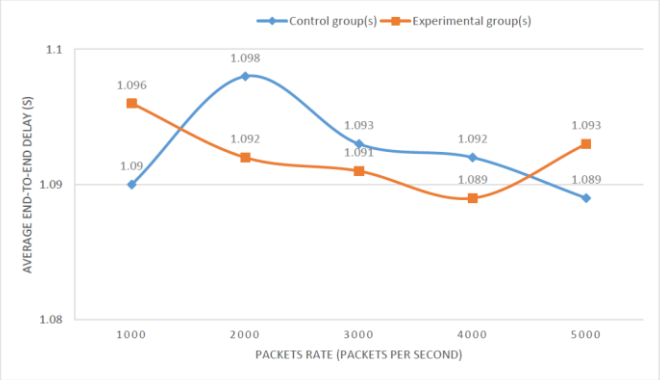

Figure 8 Average end-to-end delays with different packet rates (strategy 1)

From Figure 8, it is observed that the average delay experienced by packets of the proposed method is very close to the compared method. The differences are between 0.002~0.006.

Strategy 2 Performance Analyses

For the second strategy, LSTM algorithm classifies network situation into four categories and then determines the optimal thresholds based on these four categories. The D-ITG packet generator, network topology links bandwidth and the performance analysis are same as strategy 1. The percentage of packet losses with different packet rates observed in the second simulation are listed in Table 6.

Table 6 Packet losses in percentage with different packet rates (strategy 2)

Number of packets sent (pps)	Control group (packet loss %)	Experimental group (packet loss %)
1000	26.89	19.45
2000	27.12	20.38
3000	27.66	22.88
4000	27.59	22.95
5000	27.83	22.43

It is observed that the proposed framework provides lesser packet loss than the compare framework. The proposed framework considerably reduces the number of packets dropped. The percentages of packet losses with varying packet rate are demonstrated in Figure 9.

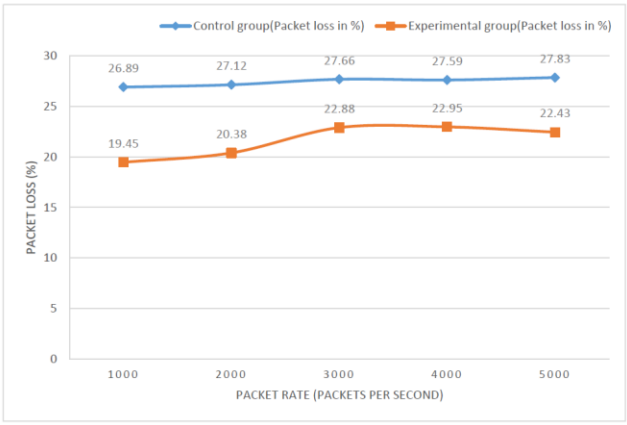

Figure 9 Packet losses with different packet rates (strategy 2)

The overall performance is similar to the result of strategy 1. Ours is much better than the control group. Throughputs observed in the second simulation with different packet rates are indicated in Table 7.

Table 7 Throughputs in Mbps with different packet rates (strategy 2)

Number of packets sent (pps)	Control group (Mbps)	Experimental group (Mbps)
1000	6.820	7.544
2000	6.831	7.438
3000	6.756	7.216
4000	6.766	7.245
5000	6.733	7.189

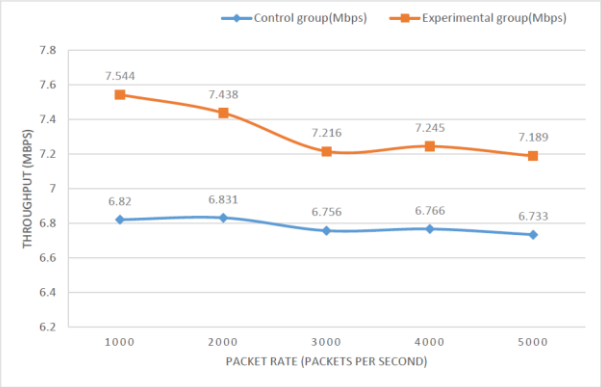

Figure 10 Throughputs with different packet rates (strategy 2)

It is observed that our method reports much better throughputs than the control group. Figure 10 depicts the variation of average throughputs with varying packet rates for two schemes.
The average end-to-end delays with varying packet rates in the second simulation is shown in Table 8.

Table 8 Average end-to-end delays in second with different packet rates (strategy 2)

Number of packets sent (pps)	Control group (s)	Experimental group (s)
1000	1.090	1.060
2000	1.098	1.088
3000	1.093	1.176
4000	1.092	1.156
5000	1.089	1.038

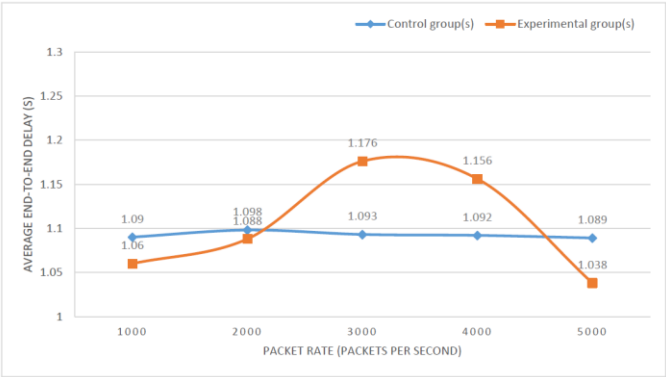

Figure 11 Average end-to-end delays with different packet rates (strategy 2)

From Figure 11, it is observed that the delays provoked by packets of our proposal are similar to the compared framework. The maximum delay does not exceed 1.2 sec and the differences are between 0.01~0.083.

CONCLUSIONS

In this paper, the LSTM algorithm is adopted to enhance QoS parameters of SDN networks. We use old network data to train the LSTM model. The forget layer in LSTM that removes non-relevant data and the candidate layer holds possible values for future adding to the cell state. The input layer decides what data from the candidate should be added. After computing the forget layer, candidate layer and the input layer, the cell state is calculated using those vectors and previous cell state. Then the output is computed.
To evaluate our proposal, we use the 4-pod fat tree topology and design two different strategies. The first one is using three categories and the other is considering four groups. The simulation results indicate that the proposed framework reduces packet loss and improves throughput while comparing to the method with fixed thresholds. The average end-to-end delays may be a little bit longer for our scheme due to the decisions of classification but the differences are small and acceptable.

REFERENCES

[1] Awduche, D. O. & Jabbari, B. Internet traffic engineering using multi-protocol label switching (MPLS). *Computer Networks,* 2002, 40, 111-129.

[2] Braun, W. & Menth, M. Software-defined networking using OpenFlow: Protocols, applications and architectural design choices. *Future Internet,* 2014, 6, 302-336.

[3] Chang, L. & Wu, Z. Performance and reliability of electrical power grids under cascading failures. *International Journal of Electrical Power & Energy Systems,* 2011, 33, 1410-1419.

[4] Chen, A. & Kasikitwiwat, P. Modeling capacity flexibility of transportation networks. *Transportation research part A: policy and practice,* 2011, 45, 105-117.

[5] Chen, M., Qian, Y., Mao, S., Tang, W. & Yang, X. Software-defined mobile networks security. *Mobile Networks and Applications,* 2016, 21, 729-743.

[6] Leiserson, C. E. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE transactions on Computers,* 1985, 100, 892-901.

[7] Li, Y. & Chen, M. Software-defined network function virtualization: A survey. *IEEE Access,* 2015, 3, 2542-2553.

[8] Mohan, P. M., Truong-Huu, T. & Gurusamy, M. TCAM-aware local rerouting for fast and efficient failure recovery in software defined networks. *IEEE Global Communications Conference (GLOBECOM)*, 2015, 1-6.

[9] Park, S. M., Ju, S. & Lee, J. Efficient routing for traffic offloading in Software-defined Network. *Procedia Computer Science,* 2014, 34, 674-679.

[10] Peterson, L. L. and Davie, B. S. *Computer networks: a systems approach*. Elsevier, 2007.

[11] Sminesh, C. N., Kanaga, E. G. M. & Ranjitha, K. Flow monitoring scheme for reducing congestion and packet loss in software defined networks. *4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2017, 1-5.

[12] Sminesh, C. N., Grace Mary Kanaga, E. & Ranjitha, K. A Proactive Flow Admission and Re-Routing Scheme for Load Balancing and Mitigation of Congestion Propagation in SDN Data Plane. *International Journal of Computer Networks & Communications*, 2019, 10 (6), 117-134.

[13] Tian, Y. & Pan, L. Predicting short-term traffic flow by long short-term memory recurrent neural network. *IEEE international conference on Smart City/SocialCom/SustainCom*, 2015, 153-158.

[14] Zhang, A., Lipton, Z. C., Li, M. & Smola, A. J. Dive into Deep Learning, Release 0.7, 2019.