A CNN-Based Routing Scheme for Minimizing TCP Flow Completion Time in SD-DCNs

Yingjie Zhou, Mingchun Xu, and Yu Chen
National Engineering Lab for Mobile Network Technologies
Beijing University of Posts & Telecommunications, Beijing, P.R.China
{yingjiezhou, mingchunxu, yu.chen}@bupt.edu.cn

***Abstract*—Data centers play a vital role in supporting the increasing demand from the six-generation (6G) networks for cloud services, big data, artificial intelligence and other data-intensive tasks. Software-defined data center networks (SD-DCNs) represent a natural evolution of traditional data center architectures aimed at improving network utilization. In this paper, we consider routing optimization for minimizing the completion time of transmission control protocol (TCP) flows in SD-DCNs. Specifically, we propose a routing scheme based on convolutional neural networks (CNN). Under different traffic transmission scenarios in SD-DCNs, the simulation results show that the flow completion time (FCT) of our scheme is much shorter compared to the equal-cost multipath (ECMP) and the shortest path first (SPF) algorithm, particularly in scenarios involving high-demand applications.**

***Index Terms*—Software-defined data center networks, flow completion time, CNN, routing.**

I. Introduction

The six-generation (6G) cloud services have wide applications including industries, education, cities, internet of things (loT) devices, and beyond [1]. Moreover, data centers provide an essential and physical infrastructure for six-generation (6G) cloud services [2]. To ensure high quality of service (QoS) in 6G cloud services, it is possible to integrate software-defined networking (that is a new network architecture capable of enhancing data transmission speed and bandwidth utilization) with data center networks (DCNs) as a natural evolution of traditional DCNs [3]; and such an integration is termed SD-DCNs and has been studied in [4].

A significant portion of traffic in data centers is generated by time-sensitive online applications such as social networking and instant messaging [5]. To provide low-latency service delivery for these applications, researches focusing on the flow completion time (FCT) of TCP flows have been conducted. Marco et al. [6] used a recursive model to derive the completion time of short-lived TCP flows by decomposing it into connection and transmission. Luan et al. [7] employed a semi-markov process to model the evolution of the TCP congestion window and deduced the flow completion time (FCT) distribution based on the queueing theory. Based on the theoretical analysis, the FCT provisioning can be achieved through optimized routing, which can be implemented in SD-DCNs.

Routing optimization in SD-DCNs mostly used flow-based routing algorithm in the SDN domain. It can be classified into two categories: 1) route based on traditional flow scheduling; and 2) route based on artificial intelligence (AI). Dhanya et al. [8]implemented equal cost multi path (ECMP) routing on SDN-based fat tree data center using the next hop selection method, Mod-N with hashing over packet header fields and it is limited to static load balanced routing method. Zang et al. [9] utilize flow information collected from edge switches and selectively reroute some flows using segment routing technology. With the development of machine learning (ML) in SDN, Jiang et al. [10] used the LightGBM toolbox to build a pre-trained model for routing prediction. Marwa et al. [11] provided a routing optimization based on Q-learning model to balance energy consumption and QoS-requirements satisfaction. However, the above work doesn't consider the critical factors: the flow completion time and the dynamic impact of TCP congestion control mechanisms on flow rates.

In this paper, we use the AI-based routing scheme to reduce the FCT in SD-DCNs. Our contribution is listed below:

1) we proposed a flow-based routing algorithm using fixed point iterations;
2) for the first time, we apply the convolutional neural network (CNN) to routing optimization for TCP applications in SD-DCNs.

To the best of our knowledge, these are not addressed before. The CNN-based scheme has two main steps: 1)it uses the flow-based routing algorithm and a simple network simulator to generate a dataset on demand matrix, while also adjusting the input demand from low to high to analogize the dynamic behevior of TCP traffic during transmission; and 2) based on this dataset, it uses CNN to build a pre-trained model to predict routes and updates the routing table periodically to adapt to different network status. Its implementation resides in the application layer of SD-DCNs.

The remainder of the paper is organized as follows. Section II presents the network model of SD-DCNs with TCP traffic. The proposed CNN-based routing scheme is introduced in detail in Section III. Simulation results and performance analysis are presented in Section IV. We summarize our work in Section V.

II. Network Model

We consider a basic SD-DCN architecture given in Fig. 1 including three layers: the data layer, the control layer and the application layer. The data layer illustrates the topology

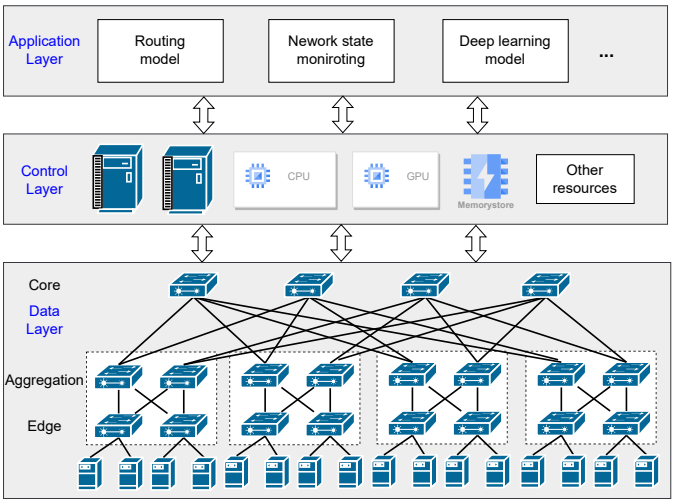


Fig. 1: SD-DCN architecture.

of fat-tree with $k = 4$, which consists of k pods with k-port switches. Each pod is composed of two (aggregation and edge) layers of $\frac{k}{2}$ switches. The network is modelled a directed graph $G(\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ is the set of SDN switches and $\mathbb{E}$ is the set of possible links between the switches through available ports. The ordered pairs (v_i, v_j) represents the link from v_i to v_j. Each link (v_i, v_j) has a capacity c_{ij}. A TCP flow from v_i to v_j over a path $p_{ij} \subset \mathbb{V}$ is f_{ij}. Let $\mathbb{F}$ be the set of K flows. Each flow $f_k \in \mathbb{F}$ is defined by the tuple (s_k, d_k), where s_k represents the source host, and d_k represents the destination host.

The aim of our solution is to find all routes $\mathbf{P} = \{p_{f_1}, p_{f_2}, ..., p_{f_K}\}$ to minimize flow completion time where p_{f_k} is the set of routes composed of active links belonging to the flow f_k. Flow completion time is affected by the TCP flow's size and TCP transmission process. The number of the packets is `flow_size/mss` and `mss` is the maximum segment size of TCP. Based on [7], we model the congestion avoidance behavior of TCP in terms of "rounds". The state transition diagram of TCP (Tahoe) congestion control is shown in Fig. 2. The rounds of TCP flows depend on the congestion window in three states: 1) slow start, 2) congestion avoiding and 3) fast recovery. The flow completion time T is:

$$T = \sum_{i=1}^{m} T_i, \tag{1}$$

where m is the number of rounds and T_i is the transfer time of the i^{th} round.

The transfer time of the round consists of the time of consecutive multiple hops on the same path, and if multiple flows pass through a switch together, the transfer time of the round is also determined by other flows. Consider the virtual network simulator with a node set $\mathbb{N} = \{1, 2, 3, \cdots, n\}$, and $\mathbb{N} = \mathbb{V}$ because the controller has the complete topology information of the data layer. Assume that a traffic flow from node s to node t over a path $p_{s,t} \subset \mathbb{N}$. The transfer time in

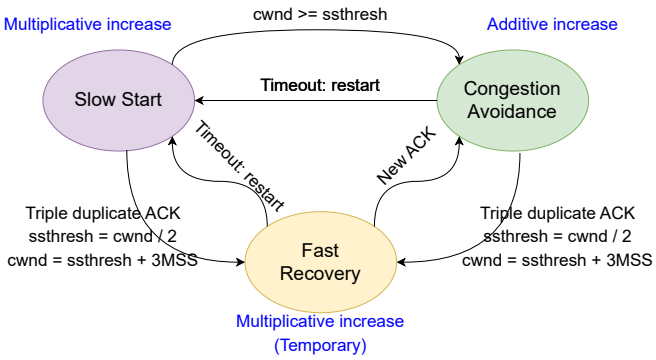


Fig. 2: State transition diagram of TCP congestion control.

the i^{th} round is a sum of the transmission delay and queueing delay along the path $p_{s,t}$:

$$T_i = \sum_{k \in p_{s,t}} \left(\sum_{w=1}^{W} s_k + q_k \right), \tag{2}$$

where W is the current size of TCP congestion window (also means the number of packets in the round), s_k is the transmission delay of the packet and q_k is the queueing delay at the k^{th} node. So the flow completion time can be expressed as:

$$T = \sum_{i=1}^{m} \sum_{k \in p_{s,t}} \left(\sum_{w=1}^{W} s_k + q_k \right). \tag{3}$$

Consider a path-edge matrix $\mathbf{A}$ ($a_{ij} \in \{0, 1\}$), i.e., for the flow j, if link i is chosen, then $a_{ij} = 1$, otherwise $a_{ij} = 0$. Therefore, $\mathbf{A}$ is a zero-one matrix. γ is the K-dimension vector and γ_j is the arrival rate of the flow j. Let λ_{uv} denotes the total arrival rate of the link (u, v) and $\bar{L}$ denotes average packet length. d_j is the FCT of the flow j. We then formulate the minimum flow completion time routing problem as a multi-objective optimization:

$$\begin{aligned} &\min_{\mathbf{A} \in R^{V \times K}} \left(d_1(\mathbf{A}\gamma), d_2(\mathbf{A}\gamma), ..., d_k(\mathbf{A}\gamma)\right) \\ &s.t. \qquad \lambda = \mathbf{A}\gamma \\ &\qquad \textstyle\sum_{(u,v) \in E} \lambda_{uv} = \sum_{(v,z) \in E} \lambda_{vz} \\ &\qquad \lambda_{uv} \bar{L} < c_{u,v}. \end{aligned} \tag{4}$$

To solve the above problem, we proposed a CNN-based routing scheme and it will be introduced in detail in Section III.

III. CNN-Based Routing Scheme

Our proposed routing scheme resides in the application layer of Fig. 1; and as shown in Fig. 3, it has two main modules: 1) routing and 2) CNN. The modules operate in the following two steps:

1) **Step 1**: the routing module uses a built-in virtual network simulator and a routing algorithm that minimizes the flow completion time to generate labeled data based on the topology monitoring; the labeled data is then stored in a database;

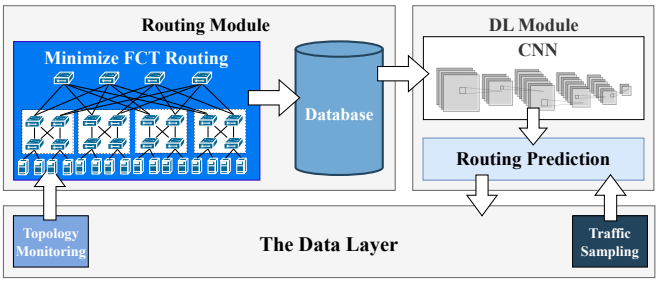


Fig. 3: Proposed routing scheme.

2) **Step 2**: the CNN module trains those data for the correspondence between the demand matrix and the routing table; the resultant prediction model is employed for routing computation in SD-DCNs.

The detailed explanation of the modules is given in Sections III-A and Sections III-B, respectively.

A. Routing Module: Minimizing FCT in SD-DCNs

The transfer of each round can be analyzed based on the queueing theory and network flow theory. The virtual network simulator uses a discrete-event simulator that has the identical network topology of the SD-DCN. The network can be modeled as a queueing network due to SDN switches' store-and-forward mechanism. Based on [7], we assume that in network scenarios with a large number of flows, the arrival of packets follows the Poisson distribution and the first-in-first-out (FIFO) discipline is used, then each queue of the target network can be modeled as an $M/D/1/FIFO$ system. The average delay of an $M/D/1/FIFO$ system is:

$$D = \frac{1}{\mu} + \frac{\rho}{2\mu\left(1-\rho\right)}, \tag{5}$$

where $\rho = \lambda/\mu$, λ is the total arrival rate of the link and μ is the average service rate. We use (5) to calculate the transmission delay and queueing delay of (3). The completion time of TCP flows can be expressed as:

$$T = \sum_{i=1}^{n} \sum_{k \in p_{s,t}} \left(\sum_{w=1}^{W} \left(\frac{1}{\mu_k} \right) + \frac{\rho_k}{2\mu_k\left(1-\rho_k\right)} \right). \tag{6}$$

We applied the algorithms in [10], [12] to develop an iterative algorithm using fixed point argument method. We introduce the vector notation first:

$$\mathbf{f} = \left(\gamma_1, \gamma_2, ..., \gamma_M\right), \tag{7}$$

where $\mathbf{f}$ is the flow vector of the network, and γ_i is the external arrival rate of flow f_i. Let $\mathbf{p}^{(n)}$ denotes the path vector at the n^{th} iteration of the algorithm whose i^{th} component $p_i^{(n)}$ represents the path for f_i:

$$\mathbf{p}^{(n)} = \left(p_1^{(n)}, p_2^{(n)}, ..., p_M^{(n)}\right). \tag{8}$$

The detailed steps are shown in Algorithm 1. In the initial state, there is no payload in the network, once the first iteration is completed, the network at this time uses $\mathbf{p}^{(1)}$ to allocate flows for the second iteration. And whichever flow is being calculated will be eliminated from the network and recalculated the path. The iteration process continues until the path vector in the current iteration matches that of the previous iteration.

Algorithm 1 Flow-based Routing Algorithm Using Fixed Point Argument Method

Input: network topology $G(\mathbb{V}, \mathbb{E})$, flow vector $\mathbf{f}$;
Output: the minimum FCT path vector $\mathbf{p}$;

$\mathbf{p}^{(0)} = []$
n = 0
for $f_i \in \mathbf{f}$ **do**
  use the algorithm in [12] to get the initial path p_i for flow f_i
  $\mathbf{p}^{(n)}$.append(p_i)
  for each link $(u, v) \subset p_i^{(n)}$ **do**
    $(u, v).pps$ = $(u, v).pps$ + $f_i.pps$
    pps means packets per second
  end for
end for
repeat
  $n = n + 1$
  for $f_i \in \mathbf{f}$ **do**
    for each link $(u, v) \subset p_i^{(n-1)}$ **do**
      $(u, v).pps$ = $(u, v).pps$ - $f_i.pps$
    end for
    for each link $(u, v) \subset E$ **do**
      $(u, v).pps$ = $(u, v).pps$ + $f_i.pps$
    end for
    use (6) to calculate the delay as link's weight
    use the algorithm in [12] to get the path p_i for flow f_i
    for each link $(u, v) \not\subset p_i$ **do**
      $(u, v).pps$ = $(u, v).pps$ - $f_i.pps$
    end for
    $p_i^{(n)}$ = p_i
  end for
until $\mathbf{p}^{(n)} == \mathbf{p}^{(n-1)}$
$\mathbf{p} = \mathbf{p}^{(n)}$

The major drawback of Algorithm 1 is that it only works for a preknown demand matrix, while the aim of our scheme is to route the data layer dynamically. To avoid the problem, we use deep-learning to assist routing. The routing module creates a unique network situation dataset for training in the CNN module. In this work, we consider different network loads to simulate TCP flows changing with the window size during transmission, then simulate the target network using different demand matrices from light to high network loads (i.e., from $\rho = 0.1$ to $\rho = 0.9$) to create the dataset. This extensive dataset serves as the foundation for the CNN module.

B. CNN Module: CNN-Based Routing Table Prediction

The architecture of CNN module is presented in Fig. 4, mainly consisted of three parts: convolution layers, pooling layers and fully connected layers [13]. The training process can be divided into two value propagation steps: the forward propagation and the backward propagation. The forward propagation is to transfer input data from the input layer to the output layer, generating the predicted results, while the backward propagation utilizes the predicted results to adjust the weights and biases of the CNN. The CNN module is thoroughly described as follows.

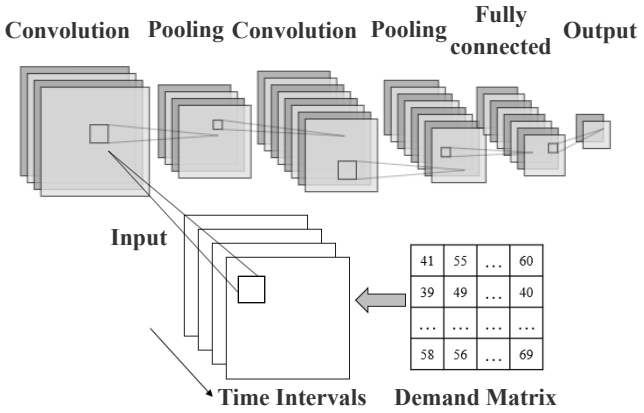


Fig. 4: The architecture of CNN module.

1) $Matrix\ Input$: We select the demand matrix in Fig. 4 as the input. It can be denoted as a two-dimensional matrix $\mathbf{X}$, where $\mathbf{X}_{ij}$ is the packet generation rate of the flow from the source node i to the destination node j. Take the scenario in fig. 3 as an example. $\mathbf{X}$ is set to a 16×16 matrix and can be obtained in SDN controller, including flows between all hosts.

2) $Convolution\ Layers$: The convolution opreation is to extract the distinguished features of the input, of which the parameters (weights and biases) consist of a set of learnable filters. Let W_l be a $M \times N$ matrix to denote the filter and the obtained feature map can be shown as follows [14].

$$\begin{aligned} x_{i,j}^{l} &= (X^{l-1} * W_l)(i,j) + b^l \\ &= \sum_{m=1}^{M} \sum_{n=1}^{N} W_{m,n} a_{i+m,j+n}^{l-1} + b^l, \end{aligned} \tag{9}$$

$$a_{i,j}^{l} = f(x_{i,j}^{l}), \tag{10}$$

where x^l is the output feature of the demand matrix at the l^{th} layer and X^{l-1} is the input feature matrix of the l^{th} layer. $f(\cdot)$ is the activation function and $a_{i,j}^{l}$ is the activated value of the unit in the i^{th} row and j^{th} column of the feature map. b^l denotes the bias of the filter and is usually a single numeric value. $a_{i+m,j+n}^{l-1}$ is the activated value of unit in the $(i+m)^{\text{th}}$ row and $(j+n)^{\text{th}}$ column. The most commonly used activation function is the rectified linear unit (ReLU) function.

3) $Softmax\ Output$: The output layer of CNN adopts Softmax as the classifition function to realize routing selection in DCNs. Each neuron in the layer contains an excitation function $p_i(z)$:

$$p_i(z) = \frac{e^{z_i}}{\sum_{j=1}^{m} e^{z_j}}, \tag{11}$$

$$z_i = w_i x + b, \tag{12}$$

where $\sum_{i=1}^{m} p_i(z) = 1$. The loss function of Softmax is denoted as LOs:

$$LOs = -\frac{1}{n} \sum_{i=1}^{n} y_i (\ln a) + (1 - y_i) \ln (1 - a), \tag{13}$$

$$a = \frac{1}{1 + e^{-x}}. \tag{14}$$

We use one-hot coding to represent the output label in the train dataset. The target routing table is marked as 1, and the others are 0. For example, when the load ρ is equal to 0.1, the first routing table will be selected and it is coded as (1, 0, ..., 0). After training, input a demand matrix and the Softmax output is a vector such as (0.8, 0.1, 0.05, ...). The maximum probability is 0.8, indicating that the routing table is table 1.

We used the dataset generated from the section III-A to train the CNN. It can be denoted as $\mathbb{D} = \{(X_k, y_k), k = 1, 2, ..., n\}, X_k \in R^{N \times N}, y_k \in R^{1 \times 9}$, where n is the number of samples. Considering the CNN training will consume a lot of computing resources, it is placed in the application layer in Fig. 3. With the training of model, each neuron gets an invariable weight. When the loss of the CNN model is low and tends to stablize, we can obtain the final model. The SDN controller will sample the network traffic as the input and utilize the trained CNN to deliver the flow table for packet forwarding in the data layer. Our scheme formulates an intelligent routing strategy and the FCT of the flows will be lower.

IV. Results and Discussion

We implemented our scheme in the application layer and evaluate the performance in an SD-DCN network simulator based on the architecture of Fig. 3. We consider the SD-DCN with a fat-tree topology: $k = 4$ in the data layer. Each link has heterogeneous bandwidth, which is randomly generated between 1Gbps and 10GMbps. Each flow has identical size of 1 MB. We simulated three traffic patterns: one-to-one traffic, one-to-all traffic and all-to-all traffic [5]:

1) **one-to-one traffic**: We generate traffic from one pod to another pod in the network. The hosts belonging to the pod send flows to all hosts belonging to another pod.
2) **one-to-all traffic**: We generate traffic from one pod to other pods in the network. The hosts belonging to the pod send flows to all hosts belonging to other pods.
3) **all-to-all traffic**: We generate traffic from each pod to other pods in the network. Each host sends flows to every other hosts from different pods.

Before we evaluated the TCP performance of our scheme, we first evaluate the performance of the CNN model in Section IV-A. In Section IV-B, we discuss the FCT performance in the SD-DCN we considered in this paper.

A. CNN Model Analysis

This section determines the structure of CNN. In order to improve the accuracy and decrease the training time, we provide 9 different CNN structures, and the detailed information

TABLE I: The detailed CNN Structure information.

No.	Input Layer	Convolution Layer (Kernel)	Pooling Layer	Convolution Layer	Pooling Layer	Fully Connected Layer	Output Layer
S1	16×16	$3 \times 3 \times 5$	2×2	$3 \times 3 \times 5$	2×2	$16 \times 16 \times 5 \times 9$	1×9
S2	16×16	$3 \times 3 \times 10$	2×2	$3 \times 3 \times 10$	2×2	$16 \times 16 \times 10 \times 9$	1×9
S3	16×16	$3 \times 3 \times 15$	2×2	$3 \times 3 \times 15$	2×2	$16 \times 16 \times 15 \times 9$	1×9
S4	16×16	$5 \times 5 \times 5$	2×2	$5 \times 5 \times 5$	2×2	$16 \times 16 \times 5 \times 9$	1×9
S5	16×16	$5 \times 5 \times 10$	2×2	$5 \times 5 \times 10$	2×2	$16 \times 16 \times 10 \times 9$	1×9
S6	16×16	$5 \times 5 \times 15$	2×2	$5 \times 5 \times 15$	2×2	$16 \times 16 \times 15 \times 9$	1×9
S7	16×16	$7 \times 7 \times 5$	2×2	$7 \times 7 \times 5$	2×2	$16 \times 16 \times 5 \times 9$	1×9
S8	16×16	$7 \times 7 \times 10$	2×2	$7 \times 7 \times 10$	2×2	$16 \times 16 \times 10 \times 9$	1×9
S9	16×16	$7 \times 7 \times 15$	2×2	$7 \times 7 \times 15$	2×2	$16 \times 16 \times 15 \times 9$	1×9

TABLE II: The accuracy of different CNN structures.

No.	S1	S2	S3	S4	S5	S6	S7	S8	S9
The Accuracy	0.9327	0.9956	0.9978	0.9674	1	1	0.9913	1	1
The Convergence Time (s)	8.54	6.93	7.20	6.48	5.12	6.64	7.49	6.21	7.36

is shown in Table I. The convolution kernel varies between 3×3, 5×5 and 7×7, and the filter of pooling layer is set 2×2. We compute the accuracy and the convergence time that are shown in Table II. We can observe that the fifth CNN structure (i.e., S5) contributes the best performance. With the increasing number and size of convolution kernels, the accuracy becomes to ascend. However, when reaching a certain level, the convergence time begins to ascend which can be caused by the increased computation complexity. Furthermore, it indicates that the modest increasing on the number and size of convolution kernels for CNN model can be conducive to the better performance. The following simulation results in SD-DCNs are reported according to the S5-based CNN structure.

B. Flow Completion Time Analysis in SD-DCNs

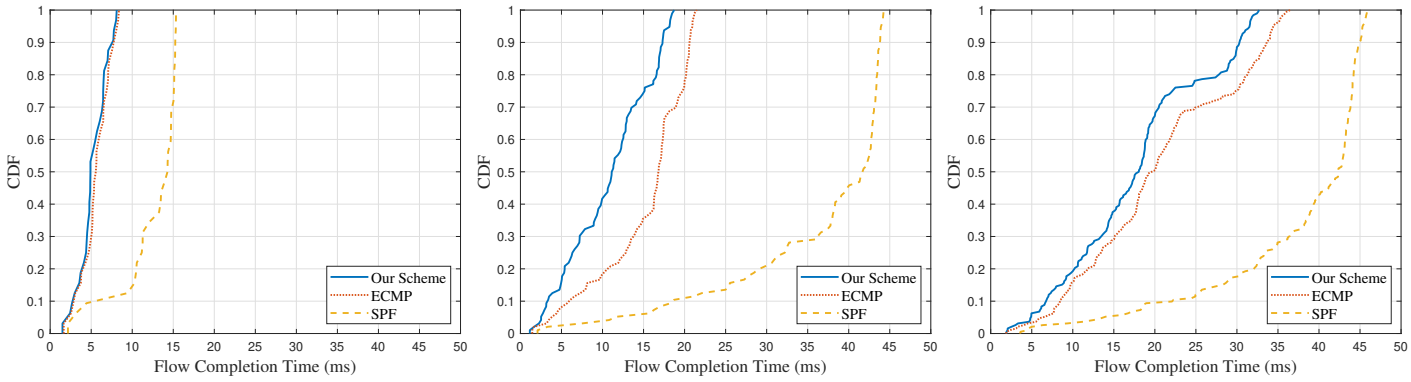
The objective of our scheme is to minimize the average flow completion time of each TCP flow in comparison with the shortest path first (SPF) algorithm and ECMP routing. Fig. 5 compares the CDF of flow completion time using our scheme with the CDF using the SPF and the ECMP algorithms under three traffic patterns. where the x-axis represents the completion time of TCP flows in seconds, y-axis shows the cumulative percentage of TCP flows. For the one-to-one traffic, 32 TCP flows are generated in the network. The completion time of the flows using our scheme and the ECMP is significantly shorter than using the SPF. The curves of our scheme and ECMP almost overlap because the traffic load is low with few flows. For the one-to-all traffic (96 flows) and all-to-all traffic (192 flows), as the number of flow increases and the window size changes, the network is more congested. It can be seen that with our scheme, the CDF of FCT is closer to the Y-axis than SPF and ECMP. This implies that more TCP flows have shorter flow completion time.

Fig. 6 shows the performance for our scheme against the SPF and the ECMP. The x-axis presents the three traffic patterns simulated in the experiments and y-axis shows the flow completion time. The performance of the SPF algorithm generally degrades when the pattern varies from one-to-one traffic to one-to-all traffic and then to all-to-all traffic. In contrast, the performance of our scheme is better than the SPF algorithm and reduces the FCT by up to 50% shorter than the SPF algorithm. For the ECMP, as the load of the network increases, the effect of our scheme is also apparent and reduces FCT by up to 21%. The reason is that our scheme is designed to minimize FCT by considering the network traffic variability in different scenarios, while the SPF and the ECMP algorithm ignores the issue on the quality of service.

Based on the above results in Fig 5-6, we may conclude that the FCT performance of our scheme is superior to SPF and ECMP, especially for the paths that have overloaded links.

V. Conclusion

In this paper, we propose a CNN-based routing scheme for TCP applications in software-defined data center networks (SD-DCNs). We first develop a flow-based minimum FCT routing algorithm and apply the convolutional neural network (CNN) to routing optimization in SD-DCNs. Based on the algorithm, we build a virtual network simulator of the real network to generate a dataset from low to high input traffic. Finally, we train the CNN model based on the dataset to predict the optimal flow path. This scheme can better choose the paths' combinations as network traffic changes. We implemented simulation at three traffic patterns: one-to-one, one-to-all and all-to-all. The simulation results show that our proposed scheme outperforms the SPF algorithm and ECMP by up to 50%, 21% respectively in terms of the flow



(a) one-to-one traffic (b) one-to-all traffic (c) all-to-all traffic

Fig. 5: The CDF of FCT under different traffic patterns.

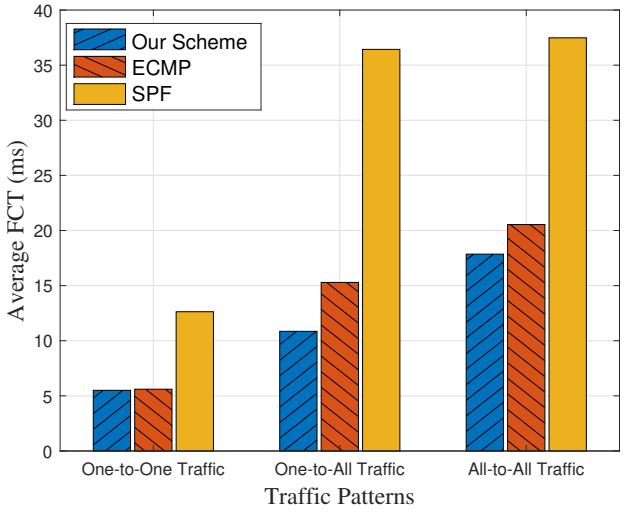


Fig. 6: Performance comparison results.

completion time (FCT). More importantly, the scheme predicts the path without prior knowledge of traffic input and provides a practical solution to routing problems in real complex SD-DCNs.

REFERENCES

[1] Ravi Teja Kamurthi, Shakti Raj Chopra, and Rahul Sharma. A burgeoning 6g technology and its cloud services. In *2022 International Conference on Emerging Smart Computing and Informatics (ESCI)*, pages 1–7, 2022.

[2] Edward Nepolo and Guy-Alain Lusilao Zodi. A predictive ECMP routing protocol for fat-tree enabled data centre networks. In *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 1–8, 2021.

[3] Deepshikha and Mayank Dave. A real-time application solution in data center networking using SDN. In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 877–881, 2018.

[4] Yuhua Xu, Zhe Sun, and Zhixin Sun. SDN-based architecture for big data network. In *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 513–516, 2017.

[5] Yingying Cheng and Xiaohua Jia. NAMP: Network-aware multipathing in software-defined data center networks. *IEEE/ACM Transactions on Networking*, 28(2):846–859, 2020.

[6] M. Mellia and H. Zhang. TCP model for short lived flows. *IEEE Communications Letters*, 6(2):85–87, 2002.

[7] Gan Luan. Estimating TCP flow completion time distributions. *Journal of Communications and Networks*, 21(1):61–68, 2019.

[8] Raj P. Dhanya and V. S. Anitha. Implementation and performance evaluation of load balanced routing in sdn based fat tree data center. In *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*, pages 1–6, 2023.

[9] Weifei Zang, Zijin Jin, and Julong Lan. An SDN based fast rerouting mechanism for elephant flows in DCN. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 363–366, 2017.

[10] Fangyi Jiang, Yingjie Zhou, and Yu Chen. Mamed: Ml-assisted minimum end-to-end delay routing in sdn-iot networks for iot monitoring. In *2023 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2023.

[11] Marwa Kandil, Mohamad Khattar Awad, Eiman Mohammed Alotaibi, and Reza Mohammadi. Q-learning and simulated annealing-based routing for software-defined networks. In *2022 International Conference on Computer and Applications (ICCA)*, pages 1–10, 2022.

[12] Y. Chen, Y. He, Z. Zhao, X. Liang, Q. Cui, and X. Tao. Demo: A quagga-based ospf routing protocol with qos guarantees. In *2018 24th Asia-Pacific Conference on Communications (APCC)*, pages 5–6, 2018.

[13] Shailender Kumar, Jitendra Kaswa, Lakshay Meena, and Mimansa Porwal. Convolution neural network-based detection in software defined networks. In *2022 2nd International Conference on Intelligent Technologies (CONIT)*, pages 1–6, 2022.

[14] Zubair Md. Fadlullah, Fengxiao Tang, Bomin Mao, Jiajia Liu, and Nei Kato. On intelligent traffic control for large-scale heterogeneous networks: A value matrix-based deep learning approach. *IEEE Communications Letters*, 22(12):2479–2482, 2018.