Forecasting SDN End-to-End Latency Using Graph Neural Network

Zhun Ge, Jiacheng Hou, Amiya Nayak
School of Electrical Engineering and Computer Science
University of Ottawa, Canada
Email: {zge084, jhou013, nayak}@uottawa.ca

2023 International Conference on Information Networking (ICOIN) | 978-1-6654-6268-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICOIN56518.2023.10048915

***Abstract*—Accurately predicting end-to-end delay is a challenging but essential problem in the networking field. By improving the delay estimation accuracy, we can distribute traffic more efficiently to enhance the networking performance future, such as satisfying packet latency requirements and improving user experience. This paper aims to predict an end-to-end delay in Software Defined Networking (SDN). We use a GNN-based model named Spatial-Temporal Graph Convolutional Network (STGCN). Benefiting the ability to capture spatial and temporal dependencies of traffic data, STGCN outperforms the baseline methodology and other three machine learning techniques, Multiple Linear Regression (MLR), Extreme Gradient Boosting (XGBOOST) and Random Forest (RF). Notably, our GNN-based methodology can achieve 97.0%, 95.9%, 96.1%, and 63.1% less root mean square error (RMSE) than the baseline predictor, MLR, XGBOOST and RF, respectively.**

***Index Terms*—GNN, SDN, End-to-end Delay Estimation**

I. Introduction

The capacity to control networks dynamically is only possible with a software-defined network (SDN). The SDN has the ability to resolve numerous challenging design issues because of its programmability and centralized control. SDN switches only have the data plane, which makes packet forwarding in the network simpler. They are configured by the logically centralized controller. The control plane in the switch is separated from the data plane, creating a logically centralized control plane with an end-to-end path-based network-wide view of packet forwarding. However, a packet delay or loss on an SDN switch can dramatically impact network performance, including Quality of Experience (QoE). Research on SDN has increased significantly in recent years, particularly in light of the proliferation of IoT devices and the potential for exploiting this flexibility to control traffic and enhance computer communications.

In order to effectively manage and operate today's increasingly complex and diversified networks, traffic conditions must be predicted. SDN traffic prediction is essential for network management and planning because it allows for the early forecasting of future traffic, which can improve network performance. The correlation and self-similarity properties of SDN network traffic can be exploited to forecast traffic and boost network performance. Because there is a finite amount of network bandwidth available, traffic classification enables us to use it as effectively as possible. Internet service providers can manage the resources by giving packets a higher priority.

A new area of deep learning is the Graph Neural Network (GNN) [1]. Utilizing deep learning and message transfer between nodes, GNNs may identify graph dependence. They are neural networks that perform operations on graphs. Strong graph representation learning capabilities have allowed GNN to show outstanding outcomes across a variety of graph data-related tasks in numerous applications. While traditional Machine Learning (ML) algorithms can predict QoS parameters rather simply, we think GNNs are far more appropriate for this use and should be researched. Prediction at the node or network level can be made quite simply because graph data can be represented either at the node level or at the level of the entire network.

In this paper, we use GNN to forecast SDN end-to-end latency. The SDN controller has a comprehensive understanding of the network and can enhance the network's routing strategy if it is aware of the latency from one node to another in advance. Our objective is to accurately estimate end-to-end delay using GNN. The GNN can be installed in an SDN controller in this scenario. As a result of the SDN controller's prior knowledge of the network architecture, the information regarding the node adjacency matrix is accessible. The SDN controller is also aware of each timestamp's end-to-end delay from one node to another. The adjacency matrix and packet delays from earlier timestamps can both be fed to the GNN by the SDN controller during the delay inference phase. The SDN controller can more effectively allot traffic after the GNN anticipates delays for the entire network at future timestamps. For instance, the SDN controller may use the Openflow protocol to provide forwarding instructions to the switches in order to plan ahead for the distribution of traffic from high-intensity to low-intensity switches.

In this paper, we use the GNN framework to predict end-to-end delay with SDN for three topologies: 15-node scale-free, 24-node GEANT2 [2] and 50-node networks. The contributions of this paper are as follows:

- We develop a line graph of the network, converting the network edges as nodes and nodes as edges.
- We apply the GNN-based model with the formulation and compare it with other machine learning (ML) models: Multiple Linear Regression (MLR), Extreme Gradient Boosting (XGBOOST) and Random Forest (RF).

We find that the GNN-based approach outperforms all ML models significantly by as much as 96.1% in the most complex

network scenario. In this study, we demonstrate the importance of explicitly including network topology in network models as well as a viable approach to utilizing GNNs to predict network traffic delays.

The rest of this paper is organized as follows: Section II overviews related work. Section III describes the proposed method concerning the data preprocessing, the STGCN framework, and the datasets we use. Section IV presents the experimentation and results. Section V concludes the paper.

II. Related work

Measurements of Quality of Service (QoS) metrics like delay, jitter, and throughput are essential for achieving QoS requirements for delay-sensitive applications. Network operators frequently use these network-centric metrics to describe network performance. Because delay-sensitive applications require high quality of service, a precise delay estimate is essential. SDN enables the collection of statistics from switches on a flow-by-flow and port-by-port basis. Traditional Machine Learning (ML) algorithms may predict QoS parameters relatively simply. The authors in [3] have described how machine learning techniques have been applied to SDNs from the perspectives of traffic classification and QoS/QoE prediction. Additionally, in order to categorize different forms of traffic in the context of SDN-IoT, the authors of [4] have investigated numerous ML models, including Random Forest (RF), Decision Trees (DT), and K-Nearest Neighbors (KNN) classifiers. According to their findings, the random forest classifier with sequential feature selection (SFS) (see [5]) approach obtains the maximum accuracy with just six features.

A traffic prediction and forecasting model for the 5G network in the SDN environment has been put forth by the authors in [6] using Gated Recurrent Units (GRU). They create their forecasting model with diffusion convolution operations in GRU to capture the spatial and temporal relationships of the features of the network traffic and employ fusion learning for long-term traffic prediction.

With deep learning models, including Convolutional Neural Network (CNN), Multilayer Perceptron (MLP), and Stacked Auto-Encoder (SAE), the authors of [7] have suggested an application-based offline and online traffic classification across a suggested SDN network testbed. The deep learning models created in the SDN controller use flow statistics data along with packet-in messages as their learning characteristics. By determining the flow's match fields, the SDN controller carries out application-based traffic classification for each flow.

For traffic classification and prediction in the context of SDN, the authors in [8] have examined some existing Machine Learning (ML) and Deep Learning (DL) techniques. The authors in [9] suggest DTPRO, a Deep Q-Network-based traffic prediction strategy, to optimize network routing efficiency. DTPRO uses traffic and congestion predictions to improve the routing configurations of conventional routing algorithms.

The authors of [10] have concentrated on predicting delays in Internet of Things (IoT) and tactile internet networks using a k-step prediction approach with NARX-enabled recurrent neural networks (RNN). Future information has been predicted using the dynamic system-representing NARX network. Using the previous and present input values, the k-step-ahead modeling time series forecasting technique predicts the following output values.

The authors of [11] have suggested a long short-term memory (LSTM)-based end-to-end delay prediction methodology. They have demonstrated that their strategy can precisely forecast future link throughputs in SDN and legacy networks. LSTM is good at predicting time-series data because it can capture the temporal dependence of the data. The authors of [12] have suggested a delay estimator based on neural networks (NNs). Various network factors have been experimented with, including topology, network size, traffic volume, and routing policies. They have demonstrated that NNs could accurately predict the typical end-to-end delay in SDN.

In [13], the authors have employed two machine learning models, RF and Neural Network (NN), to predict end-to-end delay and have offered an open simulation environment for creating new datasets with traffic generation and assessment models. They have demonstrated that end-to-end delay can be anticipated based on traffic matrix samples by considering three escalating complexity scenarios.

GNNs support the input of non-Euclidean space data, such as network topologies, knowledge graphs, and molecular structures, in contrast to CNNs and recurrent neural networks (RNNs). Numerous researchers have recently adopted graph neural networks (GNNs) for many areas. The authors of [14] have used a GNN model called RouteNet for network modelling and optimization. With GNN, RouteNet can predict Key Performance Indicators (KPIs) like mean latency and jitter per source and destination pair. The authors also suggested a GNN-based self-encoder model, named Gated Graph Auto Encoder Network (GGAE), proposed in paper [15], for precisely forecasting network delay in SDN. Recently, a GNN-based caching strategy was proposed in [16], [17]. The authors modelled the network using GNN and made node-level content caching probability predictions. They have demonstrated the outstanding performances of the GNN-based caching strategy compared with other state-of-the-art caching strategies.

In [18], the authors have proposed a model named STGCN for predicting road traffic. The STGCN model performs significantly better than other cutting-edge models since it can capture both the temporal and spatial dependencies of the input data. Our paper uses this model to extract the spatial and temporal characteristics of the end-to-end delay data in SDN. Recently in [19], the authors have used the STGCN model to predict an end-to-end delay in SDN. Their approach has achieved outstanding results, 68.5% and 78.7% better than RF and NN in the Abilene Network, respectively.

III. Method

This section introduces the methodologies used in our paper. Algorithm 1 shows essential steps of our method. Details are explained in the following subsections.

A. Data Preprocessing

Given a network topology, we consider it as a directed graph $G = (N, E)$ where graph G contains K nodes and H edges, denoted as $N = \{n_1, n_2, ..., n_K\}$ and $E = \{e_1, e_2, ..., e_H\}$, respectively. We then convert the graph into a complete directed graph $G\prime = (N, E\prime)$, where $G\prime$ contains the same set of nodes as G and K^2 edges, i.e. $E\prime = \{e_{1,1}, e_{1,2}, ..., e_{K,K-1}, e_{K,K}\}$. We can observe that $e_{i,j} \in E\prime, \forall i \in N, j \in N$ and $i \neq j$. Following that, we convert $G\prime$ into a line graph $\hat{G\prime} = (\hat{N}, \hat{E\prime})$ containing K^2 nodes and L edges, where $\hat{N} = \{\hat{n_1}, \hat{n_2}..., \hat{n_{K^2}}\}$. In order to convert $G\prime$ to $\hat{G\prime}$, each edge in $E\prime$ is converted to a node in $\hat{N}$ and $\hat{e_{i,j}} \in \hat{E\prime} \iff e_i$ and e_j share a node n_v, where $e_i \in E\prime, e_j \in E\prime, n_v \in N$.

With $\hat{G\prime}$, each node $\hat{n_i}$ is corresponding to a directed edge in the graph $G\prime$. Instead of appending end-to-end link delay to graph $G\prime$, we attach a delay into each node in $\hat{G\prime}$. This way, we convert the link-level prediction problem into a node-level prediction problem. It allows us to apply GNN in node feature prediction. It is worth noting that the line graph conversion technique achieved great performance in railway delay prediction, as suggested in paper [20].

With graph $\hat{G\prime}$, we compute its adjacency matrix, M with a size of $K^2 \times K^2$, which is defined as follows:

$$M_{i,j} = \begin{cases} 1, & \text{if } \hat{n_i} \text{ and } \hat{n_j} \text{ is connected,} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

After the graph pre-processing step, we pre-process data before feeding them into the STGCN model [18]. Our Origin-Destination (OD) traffic matrix has a uniform sampling rate of ten times per second. We pre-process data according to paper [13], where the first step is to average every 50 OD traffic matrices, i.e., all OD traffic matrices within 5 seconds. Every time we calculate the mean traffic matrices, there are 49 records overlapping with the previous step. We consider the OD traffic matrices prediction a time-series problem, and thus we use ten recent OD traffic matrices in previous time steps to predict the OD traffic matrix in the fifth future time step. Finally, we transform a batch of averaged OD traffic matrices into tensors and feed them into the STGCN model. In our paper, the batch size is 100.

B. Spatio-Temporal Graph Convolutional Networks

Recently, deep learning neural networks achieved promising results in delay estimation problems. In particular, convolution techniques are often used to extract spatial and temporal dependency of data. GNNs have also achieved outstanding results by utilizing convolution in graph-structure data [21]. GNNs have abilities to model graph data and learn node embeddings by aggregating neighbouring nodes' features.

We use a GNN-based neural network, the STGCN model, to predict end-to-end delay. STGCN model has shown advantages in node-level predictions [18]. The high-level model architecture can be found in Figure 1. Two stacked spatial-temporal convolutional blocks (ST-Conv blocks) are connected sequentially. There are two temporal gated convolutions and one spatial graph convolution in each ST-Conv block. An output layer is attached at the end of the model.

Algorithm 1 Data Pre-processing and End-to-end Delay Estimation

Input: A directed graph G, OD traffic matrices, a window length W, a batch size B, a time sequence T, a predicted time step $T\prime$, a STGCN model with parameters θ

1: Transform G into a directed complete graph $G\prime$
2: Transform $G\prime$ into a directed line graph $\hat{G\prime}$
3: Compute adjacency matrix M of $\hat{G\prime}$ using Equation 1
4: Average every W consecutive OD traffic matrices
5: Extract every T time steps' OD traffic matrices as input data X
6: Get the next $T\prime$ time step's OD traffic matrix as output data Y
7: Feed B batches of (X, Y) pairs and adjacency matrix M into two consecutive ST-Conv blocks
8: Perform gradient decent on the loss function defined in Equation 3 regarding to network parameters θ

1) Predicting End-to-end Delay

The end-to-end delay prediction is naturally a time-series problem. We use previous n time steps' OD traffic matrices to predict the OD traffic matrix in the future time step ($t_{n+N_{Future}}$), which can be mathematically formulated as follows:

$$\hat{t} = \underset{t_{n+N_{Future}}}{argmax} \ logP(t_{n+N_{Future}} | (t_0, ..., t_n)) \tag{2}$$

where $t_0, ..., t_n$ is OD flows observed in the past time series, and $\hat{t}$ is the predicted OD flows.

2) Temporal Convolution Block

As shown in Figure 1, two ST-Conv blocks sequentially extract spatial-temporal dependencies of the input data. The ST-Conv block consists of one temporal and one spatial graph convolution layer.

Each temporal convolutional layer consists of a 1-D convolutional layer with gated linear units (GLU) to bring non-linearity. The temporal convolutional layer aggregates k neighbouring nodes' features for each node in the input graph.

For each ST-Conv block, there are three convolutional layers and a sigmoid layer, which is utilized for the non-linearity. An element-wise addition is applied between input data and transformed data in the module. Afterward, an element-wise multiplication is used among computed data in previous time steps and the transformed data.

3) Loss function

We use mean squared error (MSE) to train the STGCN model, whose loss function is defined as follows:

$$L(\hat{t}; \theta) = \frac{1}{n} \sum_{i=1}^{n} ||\hat{t}(t_0, ..., t_n, M_\theta) - t_{n+N_{Future}}||^2 \tag{3}$$

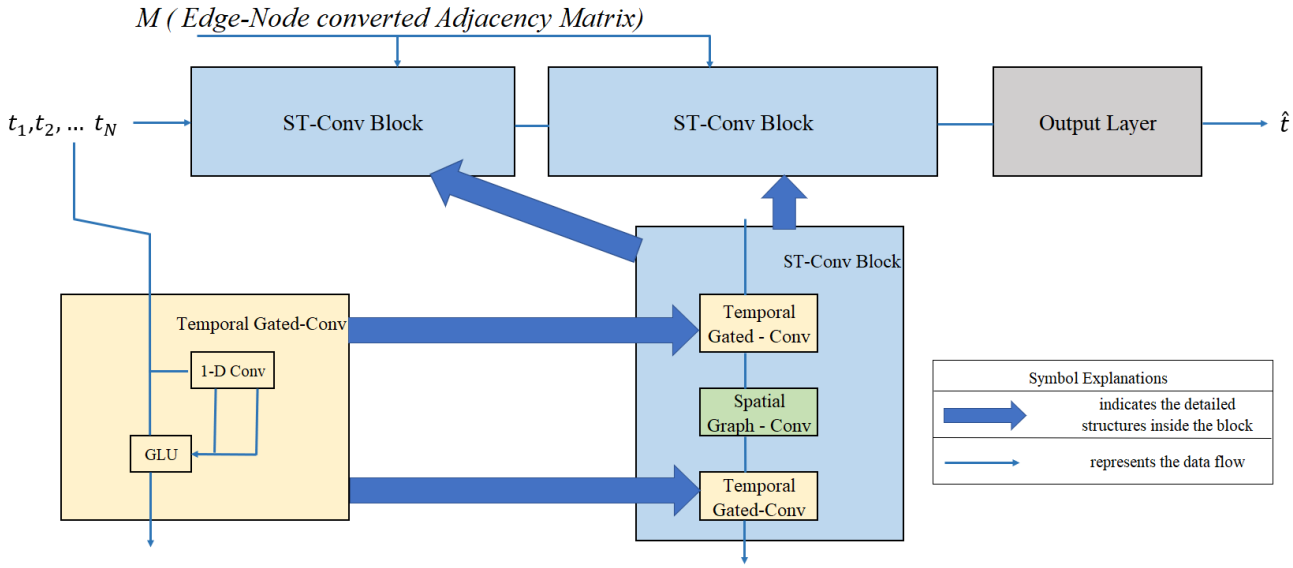

Fig. 1. STGCN Architecture: The framework includes two ST-Conv blocks and an output layer. Each ST-Conv block contains two temporal gated convolution layers and one spatial graph convolution layer. Node features in previous time sequences $t_1, t_2, ..., t_n$ and adjacency matrix M are fed into ST-Conv blocks. ST-Conv blocks extract spatial and temporal dependencies from the input data and make node-level predictions in the future time step $\hat{t}$. [19]

where θ are trainable model parameters, $t_{n+N_{Future}}$ is the ground truth OD traffic matrices, and $\hat{t}$ denotes the model's predictions.

C. Datasets

As the topology of real networks is mostly unknown, we use 15-Node Scale-Free networks as our first dataset. A scale-free network is a network whose degree distribution follows a power law, at least asymptotically. That is, the fraction $P(k)$ of nodes in the network having k connections to other nodes goes for large values of k as

$$P(k) \sim k^{-\gamma} \tag{4}$$

where γ is a parameter whose value is typically in the range $2 < \gamma < 3$ (where in the second moment (scale parameter) of $k^{-\gamma}$ is infinite but the first moment is finite) [22].

We also use another two datasets generated from RouteNet [14] using OMNeT++: one is 24-node GEANT2 [2] network in Fig 2, and another is 50-node in Fig 3.

In the dataset, each simulation contains 1000 OD traffic matrices, and 15-node, GEANT2 and 50-node networks have sizes of 15×15, 24×24 and 50×50, which contain 225, 576 and 2500 link delay values for each matrix accordingly. On top of that, each scenario we mentioned before contains 10 simulations using for average performance evaluation, and there are $2,250,000$, $5,760,000$ and $25,000,000$ values for three topologies.

IV. Simulation and Analysis

1) STGCN Model Hyperparamerters

As part of the STGCN model, a spatial kernel of size k_s = 3 and a temporal kernel of size k_t = 3 are used. Chebyshev

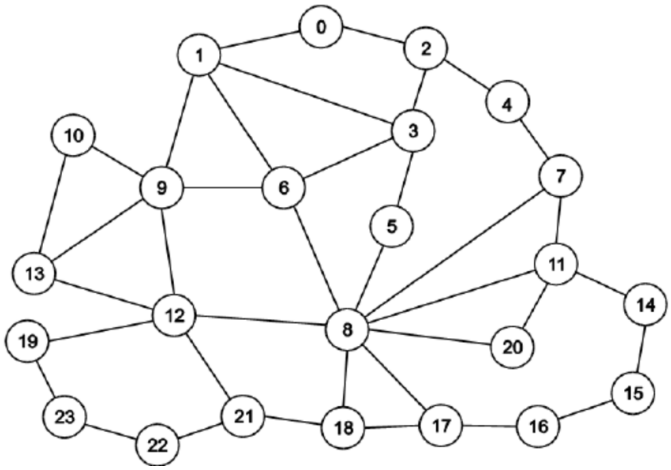

Fig. 2. GEANT2 Topology

polynomial approximation [23] is used for estimating the graph Laplacian matrix in ST-Conv blocks.

As shown in Figure 1, two ST-Conv blocks are in use. Assuming the input graph contains n nodes, each block has an output shape of $[64, 4, n^2]$. The final output $\hat{t}$ has a shape of $[n^2, 1]$. We use ReLU activation functions in the first two blocks and a sigmoid activation function in the output layer. Each model is trained using the Adam optimizer with a learning rate of 0.001 to minimize the MSE.

Before inputting data into the model, we normalize it using z-score normalization. We divide the data into $70\%/15\%/15\%$ for training, validation and testing purposes, respectively.

All training sets use NVIDIA GeForce RTX 3070 GPU with memory up to 8GB and Intel(R) Core(TM) i7-9700K

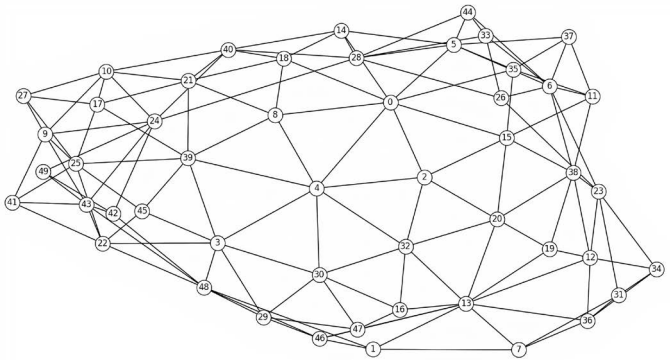

Fig. 3. 50-node Topology

@ 3.60GHz for CPU.

2) Experimental Results

We also use three other ML training methods for comparison and the average predictor as a baseline. Three methods include Multiple Linear Regression (MLR), XGBOOST (Extreme Gradient Boosting) and Random Forest (RF). We use root mean square error (RMSE) to evaluate all the models with different intensities in three scenarios.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(t_n - \hat{t}_n)^2}{n}} \tag{5}$$

Three network topologies have 15 nodes, 24 nodes and 50 nodes. Their corresponding adjacency matrices are of sizes $[255, 255]$, $[576, 576]$ and $[2500, 2500]$, respectively. We execute our experiment ten times for each situation and intensity to ensure consistent prediction and comparison. The average end-to-end delay of all OD traffic matrices are used as the baseline prediction, which is referred to as the baseline predictor. We compare the baseline predictor, several ML-based techniques, and our new GNN-based delay prediction methodology.

Table I shows the result of the baseline, MLR, XGBOOST, RF and GNN-based approach performances in three traffic topologies. We can conclude that all the ML-based approaches reach $6\%-97\%$ less RMSE compared to the baseline predictor. Particularly, XGBOOST has the worst performance in all three methods, which performs $5.9\% - 55.5\%$ better than baseline. MLR has a stable performance, which is $18.6\% - 25.2\%$ smaller than baseline. RF and STGCN give the best prediction performance among all three topologies. RF is $82.7\% - 91.9\%$ and STGCN is $94.7\% - 97\%$ better than the baseline.

Figure 4 shows the comparison between four ML methods. MLR has the biggest RMSE value in 15-Node Scale-Free topology. XGBOOST performs the worst in GEANT2 and 50-Node topology. RF has the third lowest RMSE in all topologies. Our GNN-based approach has the smallest RMSE, which performs $91.4\% - 96.1\%$ better than MLR and XGBOOST; it is also $56.9\% - 77.7\%$ better than RF, which is quite significant.

TABLE I
RMSE OF BASELINE, RF, MLR, XGBOOST AND GNN-BASED APPROACHES IN THREE NETWORK TOPOLOGIES

Method	15-Node Scale-Free	GEANT2	50-Node
Baseline Predictor	0.15937	0.27449	0.51142
MLR AVE	0.12973	0.20743	0.37759
XGBOOST AVE	0.07124	0.25820	0.396816
RF AVE	0.02738	0.03353	0.04164
GNN-based AVE	0.006103	0.0144635	0.01537

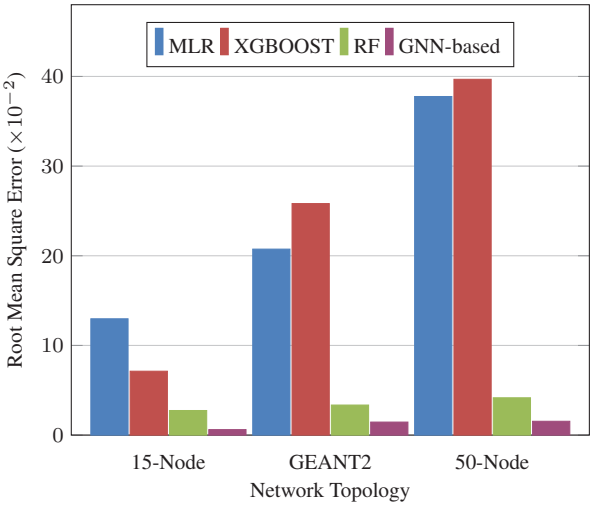

Fig. 4. Various ML methods RMSE with three network topologies: Blue bars indicate the results of MLR prediction RMSE average. Red bars represent XGBOOST RMSE average values, and green bars show RF RMSE average values and purple bars present GNN-based RMSE average results. As the network complexity increases, the RMSE increases. In all topologies, GNN-based approach has the smallest RMSE value.

We also consider mean absolute error (MAE) and weighted mean absolute percentage error (WMAPE) as the evaluation metrics, which are shown in the following equations:

$$MAE = \frac{\sum_{i=1}^{n}|t_n - \hat{t}_n|}{n} \tag{6}$$

$$WMAPE = \frac{\sum_{i=1}^{n}|t_n - \hat{t}_n|}{\sum_{i=1}^{n}|t_n|} \tag{7}$$

Table II shows the MAE and WMAPE of baseline predictor and GNN-based approaches in all three network topologies. We can observe that as the network complexity increases, the error of the baseline prediction increases a lot. However, for our GNN-based model, the error values are stable and small. Numerically, our GNN-based approach achieves 96.7%, 94.8% and 97.0% less MAE than the baseline predictor in 15-Node, GEANT2 and 50-Node scenarios, respectively. In addition, the GNN-based method performs 96.7%, 95.3% and 98.0% better than the baseline predictor according to WMAPE for three scenarios, respectively. We can find out that the GNN-based approach performs much better than the baseline

TABLE II
MAE AND WMAPE OF BASELINE AND GNN-BASED APPROACHES IN THREE TRAFFIC TOPOLOGIES

	Method	MAE AVE	WMAPE AVE
15-Node	*Baseline Predictor*	*0.1371*	*0.41554*
	STGCN	0.004552	0.01377
GEANT2	*Baseline Predictor*	*0.10996*	*0.25046*
	STGCN	0.00568	0.011867
50-Node	*Baseline Predictor*	*0.19663*	*0.29337*
	STGCN	0.005921	0.00579

predictor, and as nodes increase, the performance becomes better compared to the baseline.

In short, when network nodes rise, the baseline and other ML methods (RF, MLR, XGBOOST) performance errors increase. However, regardless of network complexity, our GNN-based delay prediction technique may deliver high accuracy. Notably, in the most complex 50-Node network environment, our GNN-based technique performs significantly better. We can infer that our GNN-based technique is capable of more correctly predicting end-to-end delay in the SDN context.

V. CONCLUSION

We used a GNN-based model to forecast the end-to-end latency of SDN in this paper. We evaluated the performance of our GNN-based delay prediction with MLR, XGBOOST, RF and the baseline predictor using three network topologies simulated from RouteNet using OMNeT++. The results show that the GNN-based technique outperforms the baseline predictor in predicting the end-to-end delay. In the best situation, our GNN-based delay prediction obtained 97.0% less RMSE, 97.0% less MAE and 98.0% less WMAPE than the baseline predictor. When compared to other ML methods, our GNN-based delay prediction technique performs the best in all testing network situations. In particular, our GNN-based model outperforms MLR, XGBOOST and RF with 95.9%, 96.1% and 63.1% reduced RMSE in the most complex 50-node network situation, respectively. Unlike other models or predictors, our GNN-based model captures both the temporal and spatial dependency of the data. Hence, our GNN-based network delay prediction model may undoubtedly be utilized in SDNs to assist controllers in better understanding network traffic circumstances and allocating traffic.

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[2] F. Barreto, E. Wille, and L. Nacamura, "Fast emergency paths schema to overcome transient link failures in ospf routing," *International Journal of Computer Networks & Communications*, vol. 4, 04 2012.

[3] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 393–430, 2019.

[4] A. I. Owusu and A. Nayak, "An intelligent traffic classification in sdn-iot: A machine learning approach," in *IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2020, pp. 1–6.

[5] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[6] K. Tamil Selvi and R. Thamilselvan, "An intelligent traffic prediction framework for 5g network using sdn and fusion learning," *Peer-to-Peer Networking and Applications*, vol. 15, pp. 751–767, 2022.

[7] L.-H. Chang, T.-H. Lee, H.-C. Chu, and C.-W. Su, "Application based online traffic classification with deep learning models on sdn networks," *Advances in Technology Innovation*, vol. 5, no. 4, pp. 216–219, 2020.

[8] A. R. Mohammed, S. A. Mohammed, and S. Shirmohammadi, "Machine learning and deep learning based traffic classification and prediction in software defined networking," in *2019 IEEE International Symposium on Measurements Networking*, 2019, pp. 1–6.

[9] E. H. Bouzidi, A. Outtagarts, R. Langar, and R. Boutaba, "Deep q-network and traffic prediction based routing optimization in software defined networks," *Journal of Network and Computer Applications*, vol. 192, p. 103181, 2021.

[10] A. R. Abdellah, O. A. Mahmood, R. Kirichek, A. Paramonov, and A. Koucheryavy, "Machine learning algorithm for delay prediction in iot and tactile internet," *Future Internet*, vol. 13, no. 12, 2021.

[11] A. Lazaris and V. K. Prasanna, "Deep learning models for aggregated network traffic prediction," in *2019 15th International Conference on Network and Service Management (CNSM)*, 2019, pp. 1–5.

[12] A. Mestres, E. Alarcón, Y. Ji, and A. Cabellos-Aparicio, "Understanding the modeling of computer network delays using neural networks," in *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, 2018, pp. 46–52.

[13] F. Krasniqi, J. Elias, J. Leguay, and A. E. Redondi, "End-to-end delay prediction based on traffic matrix sampling," in *IEEE INFOCOM Workshop*, 2020, pp. 774–779.

[14] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet: Leveraging graph neural networks for network modeling and optimization in sdn," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2260–2270, 2020.

[15] C. Zebin, W. Yichi, H. Tang, and L. Chuanhuang, "Research on intelligent perception model of sdn network delay," in *IEEE 6th International Conference on Computer and Communication Systems (ICCCS)*, 2021, pp. 452–457.

[16] J. Hou, H. Xia, H. Lu, and A. Nayak, "A gnn-based approach to optimize cache hit ratio in ndn networks," in *IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[17] ——, "A graph neural network approach for caching performance optimization in ndn networks," *IEEE Access*, vol. 10, pp. 112 657–112 668, 2022.

[18] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *arXiv preprint arXiv:1709.04875*, 2017.

[19] Z. Ge, J. Hou, and A. Nayak, "Gnn-based end-to-end delay prediction in software defined networking," *Proceedings of the 18th Annual International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 372–378, 2022.

[20] J. S. Heglund, P. Taleongpong, S. Hu, and H. T. Tran, "Railway delay prediction with spatial-temporal graph convolutional networks," in *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6.

[21] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, no. 1, pp. 1–23, 2019.

[22] J.-P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, and A.-L. Barabási, "Structure and tie strengths in mobile communication networks," *Proceedings of the national academy of sciences*, vol. 104, no. 18, pp. 7332–7336, 2007.

[23] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, 2016.