Graph Neural Network-based Delay Prediction Model Enhanced by Network Calculus

Lianming Zhang, Benle Yin, Qian Wang, Pingping Dong*
Hunan Normal University, Changsha, China
Email:{zlm,yinbenle,wangqian,ppdong}@hunnu.edu.cn

***Abstract*—Network modeling is critical to network management. Existing network modeling approaches still have room for improvement in performance prediction of important performance metrics, such as latency. Graph Neural Network (GNN) has proven their effectiveness for network performance prediction. However, the shortcomings of GNN in data interpretability limit their performance. To address this problem, we introduced network calculus to enhance the interpretation and learning of GNN for data. We proposed a modified GNN model - NetCTRT, which understands the information between topology, routing and traffic and interacts with the Network Calculus (NC)-based delay bound so that the end-to-end delay can be accurately predicted. Experimental results show that NetCTRT can significantly improve the prediction accuracy.**

***Index Terms*—Network modeling, Network calculus, Graph Neural Network, End-to-end delay**

I. Introduction

Network modeling is a very important part in the field of computer networks. The role of network modeling is to predict various network performance metrics such as latency, jitter, packet loss, etc. With the help of network models, network administrators can predict the variation of network performance metrics such as latency under "what-if" scenarios [1]. For example, when the network topology environment changes in a certain way, how the network latency will change with it. So administrators can allocate network resources more rationally, enabling efficient network planning and optimization without the need for costly on-site deployments.

In the past, efforts have been made to construct network models that can effectively predict network performance with good results. The existing network modeling approaches can be categorized into the following three types: network simulators, analytical models, and machine learning-based models [2]. Discrete-event network simulators, such as OMNET++ or NS-3 can perform accurate simulations of network packets and obtain accurate performance prediction values. However, they require high computational cost and have limited applicability for real networks. Analytical models, such as queuing theory [3] or Network Calculus (NC) [4], are widely used in network modeling. Analytical models abstract mathematical models based on certain conditions and reasonable assumptions, and eventually yield accurate prediction values. However, when the actual situation of the network is more complex, such restrictive assumptions make them difficult to describe the system accurately. Machine learning-based models show their advantages when dealing with complex systems that are challenging for analytical models [5]. Machine learning-based models are able to obtain the value of the performance metric that we want to predict based on the given input variables [6]. These approaches are not limited by the complexity of the system, but often lack a reasonable explanation of the interaction aspects of the network model and the system parameters.

Although these approaches and applications have proven to be successful, they still have their own limitations. In this paper, we propose NetCTRT, which is a network modeling approach that combines a NC-based analysis model with a Graph Neural Network (GNN)-based machine learning model. It has the mathematical theory support of analytical models, but also has the advantages of machine learning models for complex data processing, to some extent to compensate for the limitations of existing models. The approach is implemented using the RouteNet model [7] as a starting point. RouteNet relies on the message passing mechanism of the GNN model to be able to pass routing schemes in the network and abstract meaningful information about the network state [6]. GNN is well-suited for problems in communication networks. GNN not only has a strong learning capability to capture spatial information hidden in the network topology, but also has a strong ability to generalize for use in invisible topologies when the network is dynamic [8]. Although RouteNet can achieve good prediction results, its understanding of the network topology is based on a large amount of data and lacks the support of mathematical interpretation models. In contrast, there is a rigorous mathematical justification for quantitatively analyzing the system performance parameters in NC [9]. With the help of arrival curve and service curve, NC can obtain bounds on the network performance parameters, such as upper bounds on delay. In this paper, we first derive an analytical model of NC for end-to-end delay bounds based on publicly available Knowledge-Defined Networking (KDN) training datasets generated with packet-level simulator OMNeT++ [10]. Then, we implement the NC-based model inside GNN. The theoretical value of the upper bound on end-to-end delay is extracted based on the NC with the given network states in the dataset, and it is used as a new network feature quantity. This feature is then added to the input of the GNN. By adding the theoretical value of the upper bound on end-to-end delay, we believe that the theoretical support and guidance

*Corresponding author

ISBN 978-3-903176-57-7© 2023 IFIP

for the GNN can be enhanced. Experimental results show that the prediction accuracy of the improved NetCTRT model has been improved compared to the original RouteNet model and new PLNet model [11]. The main contributions of this paper are as follows:

- We determine an end-to-end delay upper bound calculation method based on NC for the publicly available KDN training datasets, and extract the theoretical value of the end-to-end delay upper bound by this method.
- We use the obtained upper bound of the NC delay together with the network topology, routing scheme, and traffic information as the input information of the GNN. The relationship between these elements is modeled inside the GNN to obtain a NetCTRT delay prediction model with better performance.
- We evaluate the implemented model and demonstrate the improvement in learning ability, prediction accuracy, and generalization ability by comparing it with existing models.

The rest of the paper is organized as follows. In Section II, we summarize the related work. In Section III, we introduce the method of NC to derive delay bounds. In Section IV, we propose a NetCTRT model combining NC with GNN. In Section V, we perform an implementation of the model and evaluate the performance of the model. In Section VI, we conclude this paper.

II. RELATED WORKS

KDN has sparked the interest of researchers, which was proposed by Mestre *et al.* [12] describes a new paradigm utilizing Software-Defined Networking (SDN), network analysis, and artificial intelligence. It advocates the use of machine learning and deep learning to gather knowledge about the network, which could be utilized to control the network using the logical centralized control capabilities provided by SDN. Inspired by KDN, researchers tried to explore machine learning models suitable for use in network modeling. Deep-Q [13] designed a quality of service inference structure using a long short-term memory model to perform inferential learning on traffic characteristics. Deep-Q achieves accurate prediction of end-to-end path delays, but fails to generalize the prediction to unknown topologies and routing schemes. RouteNet [6] utilized GNN to understand the complex relationship between topology, routing, and input traffic, resulting in accurate estimates of the average delay and jitter for each source/destination pair. In this way, RouteNet can generalize to topologies and routing schemes that are not visible in training. PLNet [11] implemented a new model by improving the link-path relationship of RouteNet. PLNet achieves good prediction accuracy and shows the the possibility of improvement and optimization of RouteNet. xNet [14] also utilized GNN for network modeling. xNet achieves a significant breakthrough in high expressiveness and fine granularity.

Several representative works mentioned above have been very successful. However, we believe that machine learning-based models always suffer from a lack of model interpretability. The combination of NC-based analysis model in machine learning can play a important role in the interpretability. By adding end-to-end delay upper bound using NC of existing data, the model can enhance the understanding of data. NC has recently been widely used in delay prediction [9], which is an effective tool for delay modeling. For example, Fantacci *et al.* [15] proposed an end-to-end delay prediction method in the context of 6G networks providing virtual reality services. Wang *et al.* [16] proposed a network evolutionary analysis model describing end-to-end delays in the tactile Internet. In recent years, GNN has also received extensive attention from researchers. GNN is particularly suitable for computer networks because of its ability to learn information such as node neighbors and their good generalization ability [17]. It has recently been applied to predict the average queuing delay [18].

In fact, DeepTMA [19] has achieved good results in combining NC with GNN by proposing a new framework. The framework predicts the best competition model and feeds back to NC for calculation by learning various series possibilities of NC. DeepTMA demonstrates the possibility and effectiveness of this combination. The main body of DeepTMA is NC-based analysis model, and GNN is only used as a tool to assist the computation, which still has the drawback of simplification for the network. In our work, NC is combined into GNN. The main body of NetCTRT is GNN, and NC is only used as a guide to enhance the understanding of the data. The model we modeled takes into account many features in the network, and learns these features through the neural network, so as to achieve accurate prediction of end-to-end delay.

III. DELAY BOUND

This subsection will introduce the definition and theorems of NC and derive the method for calculating the theoretical value of NC-based delay in NetCTRT. In order to obtain performance bounds for the flow, we need upper bounds on the flow characteristics and lower bounds on the services that the network can provide [20]. They are given by the arrival curve and the service curve, respectively.

A. (min, plus) convolution

Let f and g be two non-negative generalized increasing functions and their values can be infinite, then the (min, plus) convolution of f and g is

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t} \{f(t-s) + g(s)\}. \tag{1}$$

B. Arrive curve

The concept of an arrival curve is used to limit the amount of data sent by a flow in any time interval. A generalized increasing function α is the arrival curve of a flow R when and only when

$$\forall s \leq t, \quad R(t) - R(s) \leq \alpha(t-s). \tag{2}$$

Assuming that the flow R is bounded by the arrival curve α, then the amount of data sent by the flow is limited by $\alpha(\tau)$

for a period τ. Based on (min, plus) convolution, the definition of the arrival curve is also equivalent to

$$R(t) \leq R(t) \otimes \alpha(\tau). \tag{3}$$

The leaky bucket model is the most commonly used arrival curve model and its expression is

$$\alpha_{r,b}(t) = \begin{cases} rt + b, & \text{if } t > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where r denotes the rate of the traffic and b denotes the burst amount of the traffic. This arrival curve limits the arrival rate of the traffic to no more than r bits per second, and the burst amount to a maximum of b bits.

C. Service Curve

A service curve is an abstraction that presents the server's processing of arrival traffic. In a system S, there is an input function R and an output function R^* of traffic. A generalized increasing function β is the service curve of system S when and only when

$$\beta(t) = \begin{cases} \beta(t) = 0, & t \leq 0 \\ R^*(t) \geq \underset{s \leq t}{inf}\{R(s) + \beta(t-s)\}, & t > 0 \end{cases} \tag{5}$$

Based on (min, plus) convolution, the definition of the service curve is also equivalent to

$$R^* \geq R \otimes \beta. \tag{6}$$

Assuming that the input traffic R of system S is limited by the service curve β, then its output traffic R^* is limited by the service curve β. The service curve defines a lower bound on the services that a system can provide. The latency-rate model is the most commonly used service curve model, and its expression is

$$\beta_{R,T}(t) = R[t-T]^+ = \begin{cases} R(t-T), & t > T \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where T denotes the fixed delay, R denotes the service rate and has $T \geq 0, R \geq 0$. This service curve is usually used for approximate modeling of nodes. It denotes that the time T must be waited after the data arrive at the server to get a service with a rate greater than or equal to R.

D. Delay Bound

The upper bound on the delay can be derived with the help of the service curve and the arrival curve [21]. The delay experienced by a flow providing an arrival curve α at a node providing a service curve β is bounded by

$$d(t) \leq \sup_{t \geq 0}\{\inf_{d \geq 0}\{d : \alpha(t) \leq \beta(t+d)\}\}. \tag{8}$$

It can also be expressed as the maximum horizontal deviation of the arrival curve α from the service curve β, and its expression is

$$d(t) \leq h(\alpha, \beta). \tag{9}$$

E. End-to-end Service in Tandem

Suppose a stream passes continuously through two servers providing (min, plus) service curves β_1 and β_2 respectively, then it can be equated to this stream passing through the servers providing the service curves as

$$\tilde{\beta} = \beta_1 \otimes \beta_2. \tag{10}$$

The multi-service node analysis follows the "Pay Burst Only Once" in NC theory [20]. It refers to the fact that intermediate bursts of growth in flow of interest do not affect their performance bounds. This means that when we analyze multiple servers, we can look at them as a whole. Due to the convolutional form, the convolution of the service curve through each individual server system, i.e., a tandem system, can be treated as a single whole system [16]. Equation (10) can be generalized to the case of multiple servers. Without loss of generality, assume that a flow passes through n systems in sequence and the equivalent service curve of these n systems is

$$\tilde{\beta}_{e2e} = \beta_1 \otimes \beta_2 \otimes \cdots \otimes \beta_n. \tag{11}$$

F. End-to-end Delay Upper Bound Calculation in NetCTRT

In the topological environment of NetCTRT, we can represent the traffic flow with the help of a leaky bucket arrival curve as

$$a_f(t) = r_f t + b_f. \tag{12}$$

In the simulation, the process of transporting traffic can be considered as passing through several network systems S. We use a latency-rate model to represent the service profile of each system. Assume that there are n systems with fixed delays of $T_1, \cdots, T_N$, and their service rates of $R_1, \cdots, R_n$. The end-to-end equivalent system provides a tandem service curve expressed as

$$\beta_{\text{e2e}} = \min(R_1, \cdots, R_n) * \left[t - \sum_{i=1}^{N} T_i\right]^+. \tag{13}$$

At this point, the service rate R is greater than or equal to the average rate of traffic r_f.

The upper bound on the end-to-end delay derived from the NC can be obtained from Eq. (8) as

$$d_{e2e} = \sum_{i=1}^{n} T_i + \frac{b_f}{\min(R_1, \cdots, R_N)}. \tag{14}$$

IV. Proposed NetCTRT

A. The Framework of NetCTRT

In order to explore a more accurate network modeling method, we proposed a new GNN model based on NC called NetCTRT. NetCTRT adds NC to GNN to produce accurate prediction of end-to-end path delay. With the help of arrival curve and service curve, network performance parameters can be calculated by NC [16]. In this paper, the end-to-end delay of the system can be quickly obtained using NC. NetCTRT takes the network topology, the source-destination routing scheme (the relationship between the end-to-end paths and

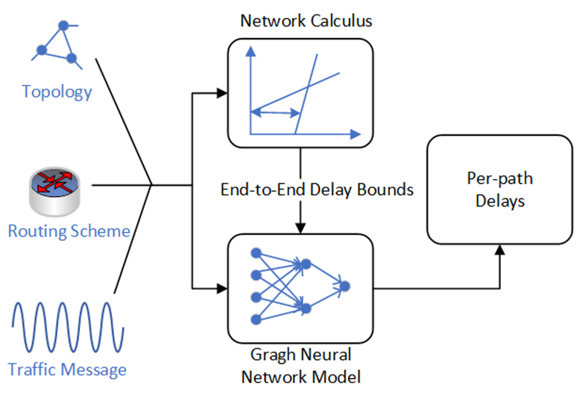

Fig. 1. The architecture of NetCTRT.

links), and the source-destination traffic matrix (the bandwidth between each node) as inputs. At the same time, based on these inputs and then based on the derivation in Section III, NC delay bounds are calculated and jointly input into GNN. For each source-destination pair, NetCTRT produces an accurate estimate of the average delay for each packet as an output. The architecture of NetCTRT is shown in Fig. 1.

NC is based on (min, plus) operation, and the process of calculating the delay has a rigorous mathematical reasoning. GNN has a good generalization ability by training and learning for a large amount of data. GNN is based on training learning for the data to get the final prediction results. It is not easy to get accurate prediction results for those network topologies that have not been trained. We think it is also important to learn the connection between many data in the training dataset. So we want to integrate NC into GNN to enhance the understanding of the network features in GNN in order to learn to get a model with better prediction accuracy and generalization ability.

B. The Principle of GNN

GNN is able to effectively propagate the information of nodes and edges to adjacent nodes or even the whole graph by converting the graph structure data into normative and standard representations and inputting them into various neural networks for training. GNN follows a neighborhood aggregation scheme, where the representation vector of a node is computed by recursively aggregating and transforming representation vectors of its neighboring nodes [22]. Let $G = (V, E)$ be an undirected graph with node $v \in V$ and edge $(v, u) \in E$. Let i_v and o_v denote the input features and output values of node, respectively. The architectural principle of NetCTRT is based on message passing neural networks [23], where the hidden representations of node h_v are iteratively passed between neighboring nodes. These hidden representations are propagated through the graph by multiple iterations until a fixed point is found. The final hidden representation is then used to predict the attributes of the nodes. This concept can be formalized as

$$h_v^t = \operatorname{aggr}\left(\left\{h_u^{t-1} \mid u \in NBR(v)\right\}\right), \tag{15}$$

and

$$o_v = \operatorname{out}\left(h_v^{t\to\infty}\right), \tag{16}$$

and

$$h_v^{t=0} = \operatorname{init}\left(i_v\right). \tag{17}$$

where h_v^t denotes the hidden representation of node v at time t. $aggr$ is a function that aggregates the set of hidden representations of neighboring nodes $NBR(v)$ of v. out is a function that converts the final hidden representation to the target value. $init$ is a function that initializes the hidden representation based on the input features.

The concrete implementations of the $aggr$ and out functions are feed-forward neural networks, with the addition that $aggr$ is the sum of per-edge terms [19], such that

$$h_v^t = \operatorname{aggr}\left(\left\{h_{NBR(v)}^{t-1}\right\}\right) = \sum_{u \in NBR(v)} f\left(h_u^{t-1}\right). \tag{18}$$

C. Message Passing of NetCTRT

In NetCTRT, the network topology is then abstracted into an undirected graph. In order to facilitate the call of the GNN on the input data, NetCTRT needs to process the data first. The path and link states are encoded separately using vectors, with the initial states being h_p^0 and h_l^0. Given a routing scheme, the link between paths and links is well represented by a GNN. By incorporating node features in the algorithm, NetCTRT simultaneously learn the topological structure of each node's neighborhood as well as the distribution of node features in the neighborhood [24]. NetCTRT predicts the path KPI by constructing a computational graph containing T iterations. Each iteration t consists of two steps: updating the path state h_p^t and updating the link state h_l^t. NetCTRT uses a Gated Recurrent Unit (GRU) [25]. The GRU that updates the path state obtains the current state of the link on the path as input. The GRU that updates the link state uses the sum of the messages from all paths containing the current link. The messages sent to the link state GRU are considered as the intermediate hidden state of the path state GRU. After the last T iterations, a readout neural network from the final path state is used to predict the path KPI. The readout neural network is implemented as a multilayer perceptron.

In NetCTRT, Algorithm 1 describes the internal architecture of the network. In this process, NetCTRT receives as input the initial path and link characteristics f_p, f_l and the routing information R and outputs the inferred predicted delay y_p. In Algorithm 1, line 2 represents the delay calculation operation of the NC to derive a new path characteristic among the information of each path present in the routing table. The loops in lines 10–12 and 15–18 represent message passing operations, which exchange encoded information (hidden state) between links and paths with each other. Line 11 uses a recurrent neural network for link-level message aggregation to deliver messages among all paths. Lines 13 and 17 are state update functions that encode the newly collected information as hidden states. The function F_p in line 21 represents a fully connected neural network modeling readout function that performs the final path-level prediction.

Algorithm 1 NetCTRT algorithm

```
Input: $f_p, f_l, R$
Output: $y_p$
1:  for each path $p$ in $R$ do
2:      $\hat{f}_p \leftarrow d_{e2e}(f_p, f_l, R)$
3:      $h_p^0 \leftarrow \left[\hat{f}_p, 0 \cdots, 0\right]$
4:  end for
5:  for each link $l$ do
6:      $h_l^0 \leftarrow [f_l, 0 \cdots, 0]$
7:  end for
8:  for $t = 1$ to $T$ do
9:      for each path $p$ in $R$ do
10:         for each link $l$ do
11:             $\tilde{m}_{p,l}^t \leftarrow \tilde{m}_{p,l}^t + RNN_t\left(h_p^{t-1}, h_l^{t-1}\right)$
12:         end for
13:         $h_p^t \leftarrow$ PathUpdate $\left(h_p^{t-1}, \tilde{m}_{p,l}^t\right)$
14:     end for
15:     for each link $l$ do
16:         $m_l^t \leftarrow \sum_{p:lep} \widetilde{m}_{p,l}^t$
17:         $h_l^t \leftarrow \text{LinkUpdate}\left(h_l^{t-1}, m_l^t\right)$
18:     end for
19: end for
20: for each path $p$ in $R$ do
21:     $y_p = F_p(h_p)$
22: end for
```

V. Performance Evaluation

A. Experimental Setup

We implemented NetCTRT using TensorFlow. In the experiment, the number of message passing iterations of the model is set to $T = 8$. Dropoutrate is set equal to 0.5, which means that each training step randomly de-activates half of the neurons in the readout neural network. This also allows us to probabilistically sample the results and to infer the estimated confidence level. During training, we minimized the Mean Square Error (MSE) between the model prediction and the true fact plus the L2 regularization loss ($\lambda = 0.1$). And we minimized the loss function using an Adam optimizer with an initial learning rate of 0.001. These parameters for NetCTRT are set by reference to RouteNet [6].

To implement the training and evaluation of GNN models, we used publicly available KDN training datasets generated with packet-level simulator OMNeT++ [10].

The dataset obtains information on end-to-end delay and packet loss between node pairs by simulating packet sending and consists of three main components: network topology configuration information, routing tables, and simulation results. The dataset is an accurate measure of the relevant end-to-end key performance indicators derived from simulations of samples with different input topologies, routing configurations, and traffic patterns. We mainly used the more classical 14-node NSF network topology [26] and 24-node Geant2 network topology [27]. In KDN training dataset, it is assumed that each node of the network can send packets to any other node. The routing table is given for each node before the start of the simulation. And a total of several routing schemes are defined by some algorithms. There are no special assumptions about routing protocols. In the simplest case, routes are computed based on the shortest path first Dijkstra algorithm.

TABLE I
Dataset information description

Parameter	Meaning
MaxPktSize	Maximum packet size of flow (bits).
ExpMaxFactor	Factor of an upper bound for exponential distributions.
Lambda	Traffic intensity.
EqLambda	Average bitrate per time unit (bits/s).
Delay	Fixed delay of this system(s).
Bandwidth	Service rate of this system (bits/s).

In the experimental setting, in addition to the end-to-end delay counted in the dataset, we also need to derive a predicted value from the NC. Without loss of generality, assuming that the flow through the network is limited by the leaky bucket model, in order to obtain the arrival curve of the flow, we need to know the average rate of the flow with the maximum burst. The parameters for generating flows in the network simulator can be obtained specifically by interpreting the information in the dataset, and the required dataset information is described in Table I.

The training and validation of the experiments in this paper are divided into two parts, the first part trains and validates the learning ability of the model, and the second part trains and validates the accuracy and generalization ability of the model.

B. Learning Ability Evaluation

First, we trained and validated with data from the NSF topology network. We trained RouteNet and NetCTRT with 240,000 training samples from the NSF network generated with a packet-level simulator. Although the samples in this dataset are all from a single topology, it contains about 100 different routing schemes and a wide variety of traffic matrices with different traffic strengths. We then performed experimental evaluation on validation samples with completely different routing schemes as well as traffic strengths from the training data.

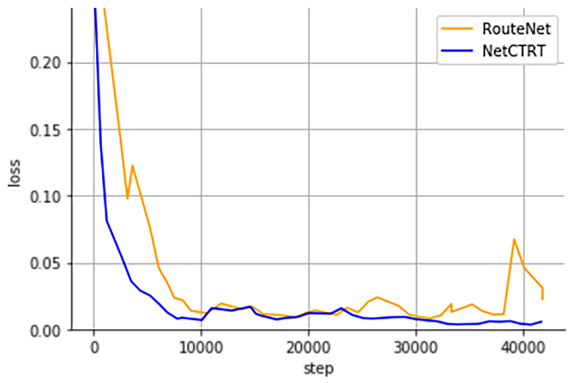

Fig. 2. Comparison of loss between the two models during training.

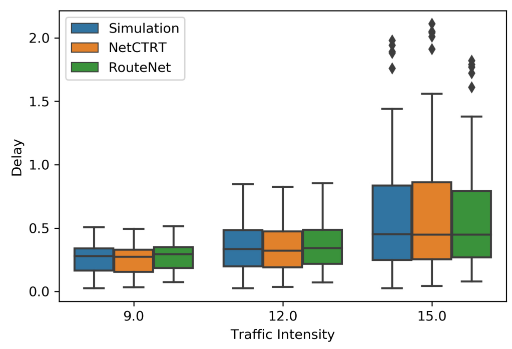

Fig. 3. Comparison between model predicted values and real simulated values.

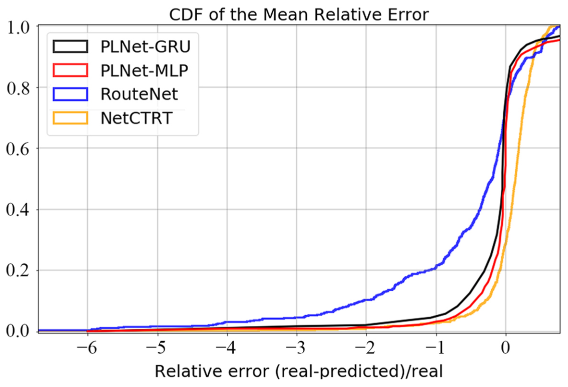

Fig. 4. Comparison of Cumulative Distribution Function of relative error of different models.

We performed more than 40,000 steps of training, with 32 randomly selected samples from the training set for each batch. Figure 2 shows the loss by Mean Absolute Error (MAE) during the training process.

It is easy to see from Fig. 2 that the training of both NetCTRT and RouteNet is stable, and the loss decreases rapidly as the training step increased. The initial loss value may be a bit too high so it is not reflected in Fig. 2. It is worth mentioning that the loss of NetCTRT decreases faster than that of RouteNet. During the training process, the losses of both models have some fluctuations, but the fluctuations of NetCTRT are obviously smaller. Therefore, it can be inferred that NetCTRT learns the latency more rapidly during the training process due to the incorporation of NC, which enhances the ability of the GNN to explain the latency. By embedding NC to calculate the upper bound on delay, NetCTRT can better understand the input data and thus achieve more accurate predictions.

To visualize the comparison between the simulated delays and the predicted delays, we present a boxplot of the predicted delays by RouteNet and NetCTRT in the NSF topology in Figure 3. The example is randomly taken from three kinds of traffic intensity verification sample sets in NSF topology that are not used for training. As can be seen, both models are trained to produce more reasonable predictions of latency. However, RouteNet's prediction of high value and low delay has obvious deviation, while the predicted value of NetCTRT is relatively closer to the real value. With the increase of traffic intensity, the predicted value distribution of NeTCTRT is closer to the simulated value than that of RouteNet. It can be seen that NetCTRT is able to achieve a better fit by enhancing the learning of the GNN for delays.

TABLE II
COMPARISON OF MODEL PERFORMANCE

Metrics	**RouteNet**	**PLNet-GRU**	**PLNet-MLP**	**NetCTRT**
MSE	3.64×10^{-4}	4.10×10^{-4}	1.76×10^{-4}	1.45×10^{-4}
R^2	0.975	0.947	0.977	0.990

C. *Accuracy and Generalization Aapability Evaluation*

In order to evaluate the accuracy of the models, we refer not only to the RouteNet model but also to two models of PLNet proposed by Kong *et al.* [11]. As shown in Table II, all four models are validated on the NSF network. Here, the MSE loss and the coefficient of determination R^2 are used as evaluation metrics. The MSE loss demonstrates the error level of the model when evaluate, and the R^2 can well describe the fit level of the model. It can be seen that the PLNet-MLP model has higher prediction accuracy compared to the RouteNet model. And the performance of our NetCTRT is the best among the four models.

From Fig. 2, we can know that the loss steadily decreases as the training steps increase and the models obtained by training tend to be stable. We obtain stable RouteNet and NetCTRT models by training, respectively. They are also evaluated in comparison with PLNet-GRU and PLNet-MLP implemented in [11]. We use samples from Geant2 network for evaluation to test the generalization ability of the models on untrained topological networks. Here, it is more convincing to visualize the probability distribution of the prediction results in the evaluation sample. Thus, we use the Cumulative Distribution Function (CDF) that reflects the relative error of the evaluation sample, as shown in Fig. 4. The distribution function shows that the prediction results of the four models were mostly distributed between -1 and 1. This shows that all these models have good generalization ability and can make predictions with small errors on invisible topological networks. The models obtained from the training are able to generalize to other networks. This is related to the use of dropout technique. It can also be seen that RouteNet showes some predictions with large errors, while PLNet-GRU, PLNet-MLP, and NetCTRT are more densely distributed with relatively small errors. In contrast, our NetCTRT model has stronger generalization ability and the prediction model can be better extended to other topological networks.

From the experimental data, it can be found that the GNN-based model proposed in this paper can model the relationship between the physical links in the network and the paths in the routing policy quite efficiently. And once the relationship between links and paths is known, the topology of the physical network itself can be ignored in the calculation. Thus the model proposed in this paper, although trained only in NSF topology, can be generalized to other topologies for more accurate predictions.

VI. Conclusion

In this paper, we built a new GNN model called NetCTRT, mainly by adding the predicted values of the analytical model NC to GNN, hoping that this combination will compensate for the lack of interpretability of some machine learning models. GNN captures the complex relationships between paths and links forming the network topology and network traffic, and NC enhances the interpretation of the relationship between end-to-end delays in paths and traffic and link information. We implemented the NetCTRT model. Then, we evaluated the model obtained from the training together with the other three models. The results show that all four models can achieve good prediction performance after training with a small number of batches under the same topology. While the accuracy of NetCTRT is slightly higher and the loss of the training process decreases faster. The NetCTRT model can provide an option for machine learning based network modeling tools. In future research, we will further explore the internal principles of GNN network modeling and also the application of NC in more complex network scenarios.

Acknowledgments

This work was supported in part by the Hunan Provincial Natural Science Foundation of China (Nos. 2022JJ30398 and 2022JJ40277) and Scientific Research Fund of Hunan Provincial Education Department of China (Nos. 22A0056 and 22B0102)

References

[1] M. B. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar, "Answering what-if deployment and configuration questions with wise," in *Proceedings of ACM SIGCOMM on Data communication*, 2008.

[2] 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.

[3] G. F. Newell, *Applications of Queueing Theory*. Wiley, 1984.

[4] J. Boudec, and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag, 2001.

[5] F. Krasniqi, J. Elias, J. Leguay, and A. E. C. Redondi, "End-to-end Delay Prediction Based on Traffic Matrix Sampling," in *Proceedings of IEEE INFOCOM WKSHPS*, 2020.

[6] K. Rusek, J. Suárez-Varela, S. Carol-Bosch, P. Barlet-Ros, and A. Cabellos-Aparicio, "Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN," in *Proceedings of ACM Symposium on SDN Research (SOSR)*, 2019.

[7] J. Suárez-Varela, S. Carol-Bosch, K. Rusek, P. Almasan, M. Arias, P. Barlet-Ros, and Albert Cabellos-Aparicio, "Challenging the generalization capabilities of Graph Neural Networks for network modeling," in *Proceedings of ACM SIGCOMM Conference Posters and Demos*, 2019.

[8] W. Jiang, "Graph-based deep learning for communication networks," *Computer Communications*, vol.185, pp: 40-54, 2021.

[9] Q. Ren, K. Liu, and L. Zhang, "Multi-objective optimization for task offloading based on network calculus in fog environments," *Digital Communications and Networks*, vol. 8, no. 5, pp: 825-833, 2022.

[10] Knowledge-defined networking training datasets. [Online]. Available: http://knowledgedefinednetworking.org/

[11] Y. Kong, D. Petrov, V. Räisänen, and A. Ilin, "Path-Link Graph Neural Network for IP Network Performance Prediction," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021.

[12] A. Mestres, A. Rodriguez-Natal, J. Carner et al., "Knowledge-defined networking," *ACM SIGCOMM Computer Communication Review*, vol.47, no.3, pp: 2-10, 2017.

[13] S. Xiao, D. He, and Z. Gong, "Deep-q: Traffic-driven qos inference using deep generative network," in *Proceedings of the 2018 Workshop on Network Meets AI & ML*, 2018.

[14] M. Wang, L. Hui, Y. Cui, R. Liang, and Z. Liu, "Xnet: Improving expressiveness and granularity for network modeling with graph neural networks," in *IEEE INFOCOM*, 2022.

[15] R. Fantacci, and B. Picano, "Edge-Based Virtual Reality over 6G Terahertz Channels," *IEEE Network*, vol.35, no.5, pp: 28-33, 2021.

[16] Q. Wang, Z. Mo, B. Yin, L. Zhang, and P. Dong, "Bounding the upper delays of the Tactile Internet using deterministic network calculus," *Electronics*, vol. 12, no. 1, Article 21, 2023.

[17] P. W. Battaglia, J. B. Hamrick, V. Bapst et al., "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[18] K. Rusek, and K. Chołda, "Message-passing neural networks learn little's law," *IEEE Communications Letters*, vol. 23, no. 2, pp: 274-277, 2018.

[19] F. Geyer, and S. Bondorf, "DeepTMA: Predicting effective contention models for network calculus using graph neural networks," in *IEEE INFOCOM*, 2019.

[20] A. Bouillard, M. Boyer and E. Le. Corronc, *Deterministic Network Calculus: From Theory to Practical Implementation*. Wiley, 2018.

[21] S. Bondorf and F. Geyer, "Virtual cross-flow detouring in the deterministic network calculus analysis," in *IFIP Networking*, 2020.

[22] K. Xu, H. Weihua, L. Jure, and J. Stefanie, "How powerful are graph neural networks?," *arXiv preprint arXiv:1810.00826*, 2018.

[23] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*, 2017.

[24] W. Hamilton, S. Z. Ying and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017.

[25] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[26] F. Barreto, E. C. Wille, and L. Nacamura Jr, "Fast emergency paths schema to overcome transient link failures in ospf routing," *arXiv preprint arXiv:1204.2465*, 2012.

[27] X. Hei, J. Zhang, B. Bensaou, and C. C. Cheung, "Wavelength converter placement in least-load-routing-based optical networks using genetic algorithms," *Journal of Optical Networking*, vol.3, no.5, pp: 363-378, 2004.