Transnet: A High-accuracy Network Delay Prediction Model via Transformer and GNN in 6G

Shengyi Ding
State Key Laboratory of Information Photonics and Optical Communications
Beijing University of Posts and Telecommunications(BUPT)
Beijing, China
dsy2022140367@bupt.edu.cn

Jin Li *
State Key Laboratory of Information Photonics and Optical Communications
Beijing University of Posts and Telecommunications(BUPT)
Beijing, China
*jinlee@bupt.edu.cn

Yonghan Wu
State Key Laboratory of Information Photonics and Optical Communications
Beijing University of Posts and Telecommunications(BUPT)
Beijing, China
kevinwu1666@bupt.edu.cn

Danshi Wang
State Key Laboratory of Information Photonics and Optical Communications
Beijing University of Posts and Telecommunications(BUPT)
Beijing, China
danshi_wang@bupt.edu.cn

Min Zhang
State Key Laboratory of Information Photonics and Optical Communications
Beijing University of Posts and Telecommunications(BUPT)
Beijing, China
mzhang@bupt.edu.cn

***Abstract*—In future 6G, bounded delay, ultra-high-reliability, and dedicated services will require high-performance network modeling techniques for the accuracy pre-validation such as network delay prediction. Recently, graph neural networks (GNNs) have been shown great potential for network delay prediction. GNNs are able to effectively capture complex topologies and node features in graph data by recursively aggregating the neighborhood information of nodes. To improve the ability to learn representations of graph data, GNNs are suitable for a variety of complex network modeling tasks with high flexibility and powerful scalability. However, the current GNN-based Routenet model can not capture the effect of the path on neighboring links, which is ineffective in complex topologies. To model the effects of the network path on neighboring links, this paper proposes a transformer-based GNN model named Transnet. In this model, the Transformer is first introduced to update the path and link states and describe the effects of the path on neighboring links based on the attention mechanism in the Transformer. Simulation results show that the delay prediction accuracy of the proposed Transnet obviously exceeds those of Routenet on multi-node topology in the Nsfnet and Synth50 datasets.**

Keywords—Network modeling, Transformer, GNN, Network delay prediction

I. INTRODUCTION

In recent years, with the commercial deployment of fifth-generation mobile networks, the research focus has been shifted to the sixth-generation networks, i.e., 6G networks [1], where bounded delay, ultra-high-reliability, and customized service provision are required for vertical industries. Therefore, how to effectively optimize future networks to maintain deterministic and dedicated network performances is an urgent problem that needs to be solved.

Network modeling is the fundamental network optimization. Traditional network modeling approaches are network calculus and queuing theory. For the network calculus modeling, the current research [2] characterizes the network traffic in terms of an envelope function, and the services are characterized by service curves. To simplify the underlying architecture, the network calculus samples path boundaries for the traffic arrivals using scale-free sampling. On this basis, simulation is performed and traffic prediction is realized. The drawback of this approach is that the architecture is too simple for the underlying layer, which sacrifices the modeling accuracy. Thus, the network dynamics are usually modeled by the interactions among network slices, system configurations, network topologies, physical infrastructure, and application traffic. Unfortunately, reflecting the intertwined relationships among the above components may be beyond the scope of the network calculus model. The advantage of queuing theory-based approaches is that they can address the problem of significant bandwidth increases and traffic growth over projections [3]. Nevertheless, queuing theory models have the disadvantage of simplifying assumptions about the underlying architecture (e.g., Poisson distribution and static routing), which is limited by the model fidelity and can not accurately reflect unpredictable network dynamics.

Recently, network modeling methods combined with machine learning (ML) have attracted a lot of attention, which has excellent representation capabilities and high computational speed in implementing network delay prediction. It is suitable for constructing accurate network performance prediction models. Because the network topology is generally expressed in the form of a graph, it is natural and reasonable to adopt graph neural networks(GNNs) for network modeling. GNN[4] is good at handling graph structure data, mainly because it can aggregate the feature information of nodes layer by layer, and gradually enrich the representation of each node through multiple rounds of information transfer, to better reflect the position and role of nodes in the graph. In addition, GNN is also able to adaptively learn the relationship weights between different nodes, which makes the model adaptable to a variety of complex graph structures and possesses strong versatility. For the GNN-based modeling, PLNet [5] used a path-link GNN model to predict the network performance distribution, RouteNet [6] adopted a GNN to estimate accurate delay for each path, and MPNN [7] predicted delay by introducing a message-passing network. However, these ML-based models with different types of GNNs may lack an understanding of the effects of the path on the interactions between neighboring links. PLNet performed a simple summation of the corresponding link features as an update of the path features without considering the effect of the interactions between the links. RouteNet applied Gated Recurrent Units (GRU) [8] for the ordered aggregation of the link features. It can only simply express the correlation between the link of the previous moment and the link of the next moment. MPNN performed vertex and edge updating

through three message-passing functions, which failed to express the connection between non-neighboring links in complex topology.

In this paper, we propose a high-accuracy delay prediction model based on transformer[9] algorithms, i.e. Transnet, to effectively express the relationship between network states and network delay. Since each link affects the path and neighboring paths composed by the link, we mainly use a transformer for path and link updating, to capture the backward and forward dependencies in the sequence of paths and links as well as the intrinsic dependencies between paths and links by the transformer encoder. The Routenet model using the Pytorch library works as the baseline in the simulation. Then the proposed Transnet is evaluated on the Nsfnet[10] dataset and the Synth50[11] dataset. The simulation results prove that Transnet is more effective and achieves much higher delay prediction accuracy in different network traffic strengths and network topologies.

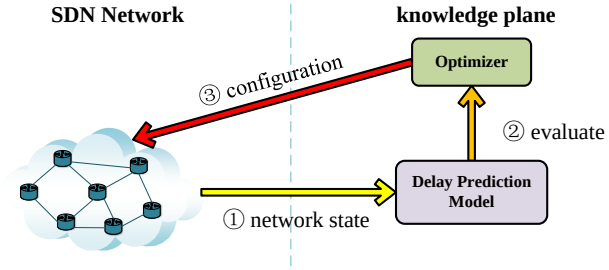

Fig. 1. Schematic diagram of network model optimization.

II. Related Work

Network modeling techniques are vital for a wide range of communication functions. Typically, simulation and analysis tools are used to construct computer network models. In our case, the network model is constructed using a neural network to capture the basic relationships between network features, e.g. traffic load, routing policy, and the resultant performance. As shown in Fig. 1, the network model can be updated with the network state (e.g., network features such as traffic matrix, bandwidth, network topology, etc.). There is an optimizer in the knowledge plane, which is defined by a given objective policy, which eventually translates into a network optimization problem[12]. We have two main requirements when choosing a model: (1) accurate performance prediction and (2) low computational cost

Since the network topology is generally expressed in the form of a graph, GNN are chosen for modeling. There are several known GNN models. A graph convolutional network (GCN) architecture for graph learning and reasoning is proposed in [13]. It performs semi-supervised learning in a direct-push setup and requires information about the entire graph during training. Rusek et al [6] proposed a novel GNN architecture to accurately predict network performance metrics including jitter, delay, and packet loss for the SDN networks, where each node is modeled as a router in an IP network. RouteNet can predict network statistics even in network topologies that were not seen during training. Moreover, PLNet [5] proposed a path-link GNN to estimate the total delay of an IP network. They modeled the IP network as a dichotomous graph since the order of links in the paths in the graph has little effect on the performance of the data flow. They compare the accuracy of the proposed method with the baseline RouteNet model and the results are similar to the baseline architecture. Furthermore, an intent-based solution for automatic network scheduling using GNN and LSTM methods is proposed in [14].

However, these ML-based models are limited to analyze the impact of each path on neighboring links. GCN suffers from scalability issues and works on a fixed graph, which is not suitable for QoS-critical and dynamically changing topology networks. Routenet assumes that all paths have the same impact on links, which is inconsistent with the actual network situation. PLNet updates all links with a simple weighted summation of their characteristics, which lacks full consideration of the impact of different links. Features are simply weighted and summed up, which lacks the consideration of the impact of different links. To capture the dependency and up-down sequence information in the sequence, we propose the Transnet use the autocorrelation property of the transformer. The advantage of Transnet over other GNN-based network models is that it introduces a self-attention mechanism that captures complex relationships between nodes and improves the ability to model long-range dependencies. Through the self-attention mechanism, Transnet can effectively capture the interactions between different links and transform them into features for state updates.

III. System Model

A. Notions

A computer network G can be expressed as $G = (P, L)$, $L = \{l_1, l_2 \dots l_i\}$ is a series of links, and i is the number of links. $P = \{p_1, p_2 \dots p_k\}$ is a series of end-to-end paths, and k is the number of paths. $R = \{r_1, r_2 \dots r_n\}$, r is a route consisting of a topological matrix of routes for a particular path. In our paper network features include path features and link features. For path features, we usually use the traffic$T = \{t_1, t_2 \dots t_i\}$to denote. Link features we use the bandwidth$B = \{b_1, b_2 \dots b_j\}$to denote. The delay of the entire model output we usually denote by $D = \{d_1, d_2 \dots d_i\}$, then the overall network characteristics can be expressed as $N = (B, T).U$ denotes the aggregation function for constructing link features into path-specific link matrices based on a particular routing matrix R.

B. Model architecture

As mentioned above, the entire network features can all be represented as path features and link features. For example, the path feature can be represented as the traffic flow through this path, and the link feature can be represented as the bandwidth of this link. Therefore, the relationship between various complex network internal features can be expressed as the relationship between the path and link of the network, which is also applied to RouteNet, and PLNet.

To better understand the effect of the path on neighboring links, we propose a novel model called Transnet based on the Routenet framework, as shown in Fig. 2: Transnet is a model that takes path characteristics, link characteristics, and routing scheme as inputs and loops over T times. The initialization phase of the Transnet model selects the traffic of each path segment as each path characteristic and uses the bandwidth of each link segment as each link characteristic. In the initialization phase, the flow rate of each path is used as each path feature and the bandwidth of each link is used as each link feature. After collecting the corresponding traffic and bandwidth data, we use a linear layer for linear embedding, which can map the traffic value of each link and

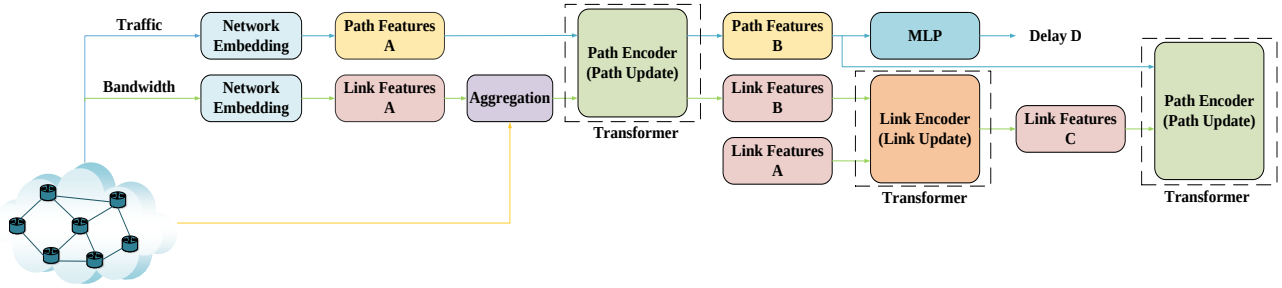

Fig. 2. Transnet structure diagram.

the bandwidth value of each link into a continuous space with low dimensions that is easy to handle. In addition, we design an aggregation layer, the aggregation layer can combine the link features with the routing matrix as a common input, The significance of this is to be able to express the relationship between the path and the links through which the path passes.

Then, Transnet employs a transformer encoder as a path update function, compared to Routenet which employs GRU as a state update function, The advantage of Transformer is that it can better capture long-distance dependencies in sequences, which makes it more suitable for modeling complex sequences, and since the transformer employs a self-attention mechanism, this allows the model to dynamically attend to information at different locations in the input sequence during learning. In contrast, GRU suffers from gradient vanishing or gradient explosion problems when dealing with complex sequences, which Transformer avoids. We take the input path features and link features as input to the transformer and output an updated path state. We similarly design a transformer encoder as a link update function which is used to update the link state. After performing T cycles of updates, we read out the updated path state through an MLP layer, and the result of the readout is the delay predicted by the model. The following Algorithm 1 explains the internal framework of Transnet and the feature update process in detail.

Algorithm1: Transnet: Transformer-Based Delay Prediction Model

Input: Network Features: $N = (B, T)$, Routing Scheme $\mathcal{R}$

Output: Prediction Delay: $\mathcal{D}$

//Initialize path features and link features

for each path **do** $p_k^0 = [t_i, 0 \dots 0]$

end

for each link **do** $l_i^0 = [b_j, 0 \dots 0]$

end

for $t = 1$ to T **do**

  for each path k in p **do** // path state update

  $\mathcal{L}_i \leftarrow U(l_i^{t-1}, \mathcal{R})$

  $p_k^t \leftarrow$Path-Transformer $(p_k^{t-1}, \mathcal{L}_i)$

  $m_{p,l}^t \leftarrow p_k^t$

  end

  for each link i in L **do** // link state update

  $l_i^0 \leftarrow$Link-Transformer $\left(l_k^{t-1}, \sum_{p:l\in p} m_{p,l}^t\right)$

  end

end

$\mathcal{D} = \text{MLP}(p_k^t)$

C. Path update process

Routenet and its related variants use GRU for path and link state updates. The use of GRU is suitable for fixed-length data, but when dealing with different network topologies, the length, and size of the input sequences are often different, so the operation with GRU requires padding or truncation and pre-processing the data to a fixed length, which introduces additional complexity and makes the computational cost higher.

To address this issue, we operate with a Transformer instead of GRU, as the Transformer's self-attention mechanism tends to allow the model to dynamically focus on different parts of the sequence data depending on the input data content while processing it. This means that the model can adaptively allocate attention based on context without the need for a fixed-length window, which allows the Transformer to capture long-distance dependencies while modeling different positions in the sequence.

Once we have extracted the relevant information in the link as well as the relevant information in the path we update it using a Path Encoder consisting of a multi-layer TransformerEncoder Layer[16], Each of these layers contains a self-attention mechanism and feed-forward neural network. These layers are used to encode the input path information and link information to capture the links between features at different locations. After encoding the path state contains updated path information which captures the relationship between features between different locations. These encoded features are integrated into the new path state to update the path representation. The specific transformer update process is described in detail below:

$$Q = \mathrm{f}_1\left(\mathrm{p}_k^{t-1}, \mathcal{L}_i\right) \tag{1}$$

$$K = \mathrm{f}_2\left(\mathrm{p}_k^{t-1}, \mathcal{L}_i\right) \tag{2}$$

$$V = \mathrm{f}_3\left(\mathrm{p}_k^{t-1}, \mathcal{L}_i\right) \tag{3}$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V = X \tag{4}$$

$$FFN(X) = RELU\left(XW_1 + b_1\right)W_2 + b_2 \tag{5}$$

$$Output = \Omega\left(X + FFN(X)\right) \tag{6}$$

The TransformerEncorder contains two sub-layers, the Multi-head Attention Layer and the Feedforward Neural Network Layer, which are integrated through residual connectivity and layer normalization. In the multi-head attention layer,Q denotes the query vector,K denotes the key vector, Vdenotes the value vector.f_1,f_2 and f_3 usually denotes the linear computational functions in terms of $p_k^{t-1}$and $\mathcal{L}_i$. In the attention head firstly the Query, Key, and Value are computed. Next, formula (4) computes the dot product of Query and Key to get the attention scores. These scores are used to measure the relevance of Query and Key. Where$\sqrt{d_k}$ denotes the dimension of the key and softmax function is used

to calculate the weights and X is obtained as input feature by calculating formula(4). In the feed-forward neural layer, $RELU$ denotes the activation function, W_1 and b_1 denotes the weights and bias of the first layer, W_2 and b_2 denotes the weights and bias of the second layer, FFN denotes the feed-forward neural network computation, and Ω denotes the layer normalization operation. The output of the feedforward neural layer is obtained by formula(5). Then the updated path state p_k^t we need is obtained by formula(6) by connecting the multi-head attention layer with the feedforward neural layer and normalizing it.

IV. Results and Discussions

In this section, we use two mainstream network datasets to validate the accuracy of the models. We reproduce the Routenet[6] model and use it as a baseline model, then we compare the delay prediction and error distributions of Routenet, PLNet, and Transnet under different traffic intensities, network topologies, and routing scenarios through extensive experiments. Our experimental results show that Transnet can produce more accurate delay prediction results.

A. Simulation setup

Our simulations use the Pytorch framework and are run on Tesla V100 GPUs. We run our experiments on the 14-node Nsfnet and 50-node Synth50 datasets from the open-source Knowledge-Defined Networking (KDN) project. The datasets are generated by a packet-level simulator, OMNet++[15], and these datasets contain simulation results of latency, jitter, and packet loss for different network topologies. The IP network scenarios considered in the dataset have the form of an undirected graph of nodes, links, adjacency matrices, routing configurations, queue statistics, and port states. Each node in the dataset represents the router and the link connecting the router. A node can send packets to any other node. A routing table for each node is also provided and routes are calculated based on the SPF (Shortest Path First) algorithm. For routing configuration, more than 100 different routing schemes are defined for each topology, and the details of the KDN dataset are shown in Table I:

TABLE I. Statistics of Network Datasets

Topology	Nodes	Maxlinks	Max Path Length	Samples
Nsfnet	14	42	5	12000
Synth50	50	280	16	29000

B. Training and evaluation

Before training, the training and validation sets in the dataset were divided into 4:1.Compared to Routenet, we increased the number of hidden layers of path and link when designing the Transnet, we set the hidden layers of path and link to 64, then for forward propagation, we used 12 message passes and set the readout layer to 8. In addition, we used the Adam optimizer and set the initial learning rate to 0.005. In addition, to measure the accuracy of the prediction, we used the following evaluation metrics for validation. Where n is the number of samples, d_i is the true delay, d_i' is the model prediction delay.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(d_i - d_i^{'}\right)^2 \tag{7}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{d_i - d_i^{'}}{d_i}\right| \tag{8}$$

MSE (Mean squared error) is the most commonly used evaluation metric to measure the actual delay against the true delay, but when the error is too small, the squared property makes the error difficult to measure, so we introduce MAPE (Mean absolute percentage error) to reduce this effect. An accurate delay prediction model will show low MSE and MAPE. Table II shows the comparison of training results for Transnet, Routenet, and PLNet. As the results of Table II show, Transnet outperforms Routenet and PLNet for both evaluation metrics we provided.

TABLE II. The Prediction Results of Two Networks

Transnet&Routenet,PLNet	Nsfnet	Synth50
MSE(10^{-4})	0.69/0.78/0.82	1.39/2.02/1.89
MAPE(10^{-2})	0.96/1.41/1.63	1.88/2.31/2.47

C. Performance of the Transnet

To demonstrate the performance of Transnet, we adopt the following two methods for illustration, Fig. 3 is the prediction error distribution graph, from the graph we can observe that the peak of the error is close to 0, which indicates that the average prediction of the model is close to the true value. Moreover, the shape of the whole graph shows a normal distribution, which indicates that the prediction errors of the model are roughly symmetrical in the positive and negative directions and are concentrated near the center. This information tells us that the model's delay prediction is close to the true value and the results are reliable.

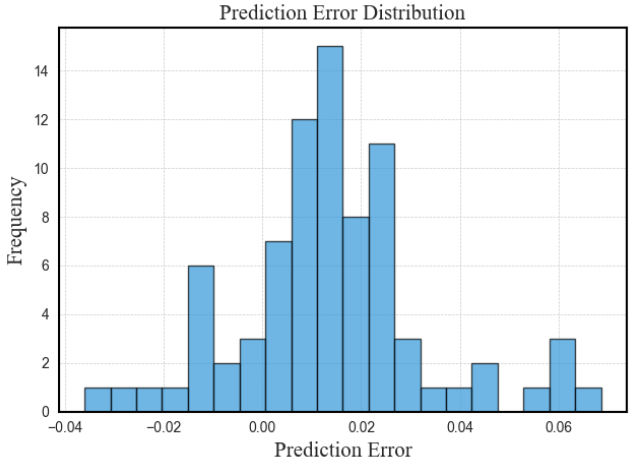

Fig. 3. Transnet error distribution diagram.

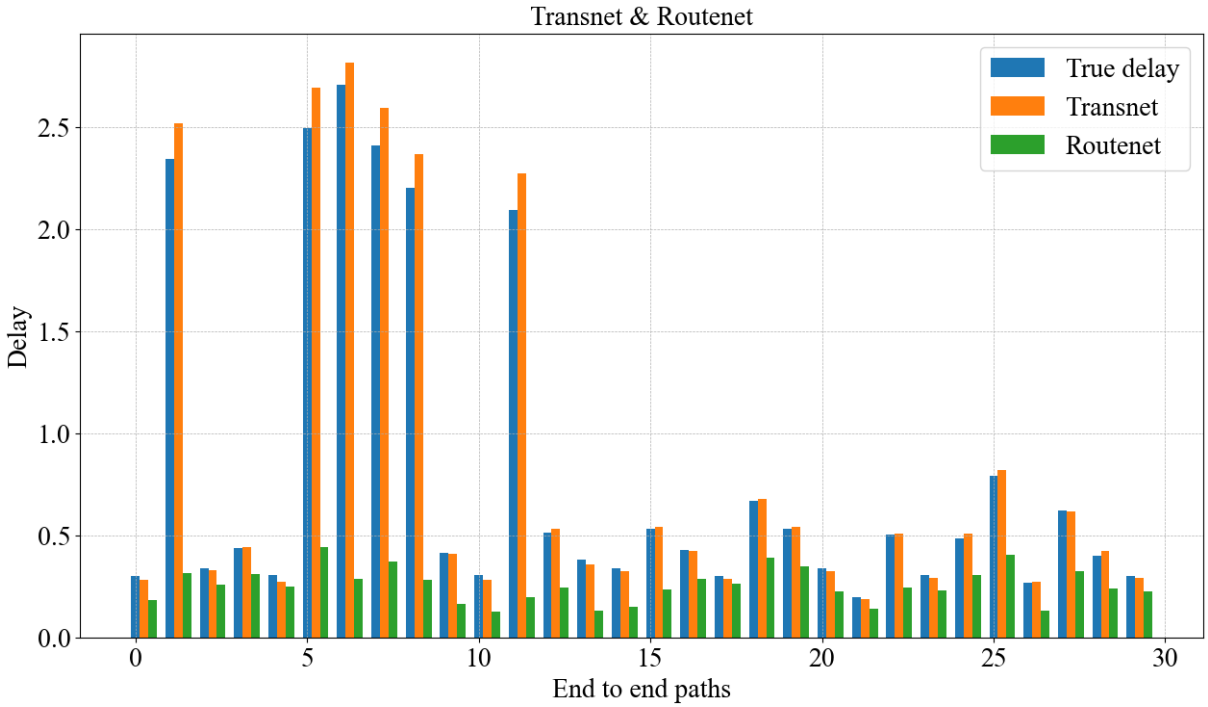

Fig. 4. Histogram of Transnet vs. Routenet prediction results.

In Fig.5, we show regression plots of the results of predicting a randomly selected topology on the Synth50 dataset in the trained Transnet model. The dots indicate the median prediction and the red dashed line indicates the true delay value. The experimental results show that the algorithm has a low overall prediction error, but the prediction error is more pronounced for high latencies than for low latencies. This is because end-to-end node pairs with high latency are rare during training because they generate large gradients, which in turn leads to the possibility that they may be considered outliers in the data during training.

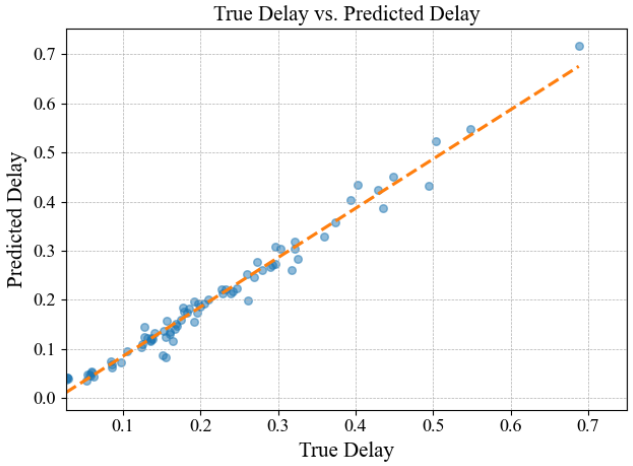

Fig. 5. Transnet regression plot.

D. *Comparison of delay prediction*

In this section, we test the models trained in the Nsfnet dataset in the Synth50 dataset respectively, and the experimental results are shown in Fig.4: the blue histogram in the following figure indicates the real experiment, the yellow histogram indicates the predicted value of Routenet, and the green histogram indicates the predicted value of Transnet, which is a test set in Synth50 that we arbitrarily take out for testing. Synth50 test set for testing, the test set selected routes have a total of 30 paths, We find that the first 15 paths in the Transnet prediction are significantly better than Routenet, in the latter 15 paths in the prediction of the Transnet and Routenet close to, but the overall effect of the Transnet still to slightly better than Routenet, which can show that Transnet can predict the results more accurately than Routenet.

In network scenarios, delay prediction models with a more concentrated distribution of prediction errors show better stability. A small number of results with large prediction errors may pose a significant security risk to network management. Moreover, these errors may not be reflected in the prediction results, so we will compare the cumulative distribution function (CDF) of the two models for comparison

$$e = \frac{d - d'}{d} \tag{9}$$

where e denotes relative error, d denotes true delay, and d' denotes prediction delay. The relative error function curve of a model with accurate prediction converges quickly to 0 for $e < 0$ and to 1 for $e > 0$. We tested the model on the validation set of the Nsfnet dataset and the validation set of Synth50, and the results of the relative error are shown in Fig.6 and Fig.7: the horizontal coordinate of the figure represents the relative error e, and the vertical coordinate represents the cumulative probability, from which we can conclude from this figure that the delay prediction results provided by Transnet work better on the Nsfnet dataset as well as the Synth50 dataset. Especially on the Synth50 dataset, which is a multi-node topology, the effect of Transnet is significantly better than that of Routenet and PLNet.

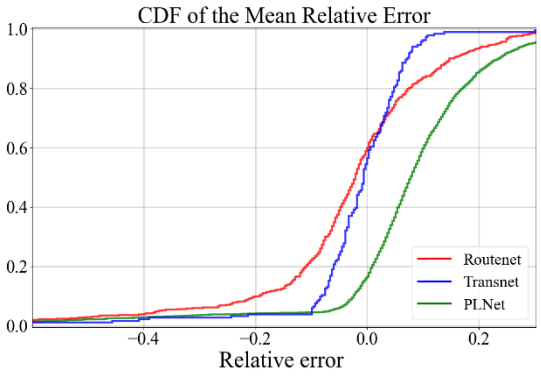

Fig. 6. The Nsfnet CDF Comparison.

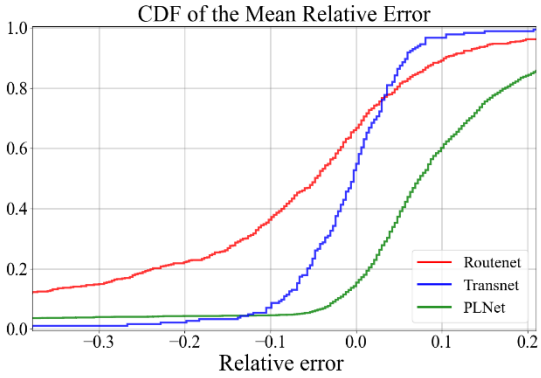

Fig. 7. The Synth50 CDF Comparison.

V. Conclusion

In this paper, we first analyze the traditional approaches to network modeling such as network algorithms and queuing theory, and analyze the characteristics. The current machine learning-based network modeling methods have been reviewed. We choose to use the Transformer algorithm instead of other classical models mainly because traditional recurrent neural networks (e.g., GRUs) usually simply aggregate these features and ignore their possible interdependencies due to the complex topology and non-local correlations between nodes in the graph data, and Transformer's self-attention mechanism can capture these features more flexibly. The Transformer has better lateral information transfer capability to handle long-distance dependencies between nodes. An end-to-end delay prediction model based on the transformer architecture has been proposed and evaluated on the open-source dataset provided by the KDN project. The results show that the proposed Transnet achieved higher prediction accuracy and is more effective than Routenet on the multi-node dataset Synth50. In future work, we will explore the effectiveness and generalization of the proposed Transnet for comprehensive metrics including the jitter, packet loss.

Acknowledgment

This work is supported in part by the National Key R&D Program of China, Grant 2022YFB2901100, and the National Natural Science Foundation of China, Grant 61975020.

References

[1] M. -I. Corici, F. Eichhorn, V. Gowtham, T. Magedanz, E. -R. Modroiu and F. Schreiner, "How Organic Networking meets 6G Campus Network Management Challenges," 2023 26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 2023, pp. 169-174.

[2] F. Ciucu and J. Schmitt, "Perspectives on network calculus: no free lunch, but still good value," ACM SIGCOMM Computer Communication Review, vol. 42, no. 4, pp. 311-322, 2012.

[3] I. Emre Telatar and R. Gallager, "Combining queueing theory and information theory for multi-access," Proceedings of 1994 IEEE International Symposium on Information Theory, Trondheim, Norway, 1994, pp. 286-288.

[4] Y. Zhu, W. Liu, S. Ling and J. Luo, "Network Modeling Based on GNN and Network Behaviors," 2022 7th International Conference on Computer and Communication Systems (ICCCS), Wuhan, China, 2022, pp. 554-559.

[5] Y. Kong, D. Petrov, V. Räisänen and A. Ilin, "Path-Link Graph Neural Network for IP Network Performance Prediction," 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), Bordeaux, France, 2021, pp. 170-177.

[6] 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," in IEEE Journal on Selected Areas in Communications, vol. 38, no. 10, pp. 2260-2270, Oct. 2020.

[7] K. Rusek and P. Chołda, "Message-Passing Neural Networks Learn Little's Law," in IEEE Communications Letters, vol. 23, no. 2, pp. 274-277, Feb. 2019.

[8] Y. Wu, Z. Zhang, C. Xie, F. Zhu, and R. Zhao, "Advancing Referring Expression Segmentation Beyond Single Image," Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2023, pp. 2628-2638.

[9] A. Vaswani, P. Ramachandran, A. Srinivas, N. Parmar, B. Hechtman and J. Shlens, "Scaling Local Self-Attention for Parameter Efficient Visual Backbones," 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 2021, pp. 12889-12899.

[10] 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.

[11] "Knowledge-defined networking training datasets," [Online]. Available: http://knowledgedefinednetworking.org/. Accessed: Sep. 10, 2023.

[12] K. Higashimori, F. Inuzuka and T. Ohara, "Physical topology optimization for highly reliable and efficient wavelength-assignable optical networks," Journal of Optical Communications and Networking, vol. 14, no. 3, pp. 16-24, March 2022.

[13] M. Ghorbani, M. S. Baghshah and H. R. Rabiee, "MGCN: Semi-supervised Classification in Multi-layer Graphs with Graph Convolutional Networks," 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Vancouver, BC, Canada, 2019, pp. 208-211.

[14] T. Ahmed Khan, K. Abbas, J. J. Diaz Rivera, A. Muhammad and W. -c. Song, "Applying RouteNet and LSTM to Achieve Network Automation: An Intent-based Networking Approach," 2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS), Tainan, Taiwan, 2021, pp. 254-257, doi: 10.23919/APNOMS52696.2021.9562499.

[15] A. Varga, "Using the OMNeT++ discrete event simulation system in education," IEEE Transactions on Education, vol. 42, no. 4, pp. 1-11, Nov. 1999.

[16] K. Hu, H. Yang, Y. Jin, J. Liu, Y. Chen, M. Zhang, and F. Wang, "Understanding User Behavior in Volumetric Video Watching: Dataset, Analysis and Prediction," Proceedings of the 31st ACM International Conference on Multimedia, 2023, pp. 1108-1116.