FLOWDT: A FLOW-AWARE DIGITAL TWIN FOR COMPUTER NETWORKS

Miquel Ferriol-Galmés[*], *Xiangle Cheng*[†], *Xiang Shi*[†], *Shihan Xiao*[†],
Pere Barlet-Ros[*], and Albert Cabellos-Aparicio[*]

[*] Universitat Politècnica de Catalunya, Spain, [†] Huawei Technologies, China

ABSTRACT

Network modeling is an essential tool for network planning and management. It allows network administrators to explore the performance of new protocols, mechanisms, or optimal configurations without the need for testing them in real production networks. Recently, Graph Neural Networks (GNNs) have emerged as a practical solution to produce network models that can learn and extract complex patterns from real data without making any assumptions. However, state-of-the-art GNN-based network models only work with traffic matrices, this is a very coarse and simplified representation of network traffic. Although this assumption has shown to work well in certain use-cases, it is a limiting factor because, in practice, networks operate with *flows*. In this paper, we present *FlowDT* a new DL-based solution designed to model computer networks at the fine-grained flow level. In our evaluation, we show how *FlowDT* can accurately predict relevant per-flow performance metrics with an error of 3.5%, FlowDT's performance is also benchmarked against vanilla DL models as well as with Queuing Theory.

***Index Terms*—** Network Modeling, Machine Learning, Graph Neural Networks

1. INTRODUCTION

Network modeling is a core component of network control and management. Particularly, models allow network researchers and administrators to explore new protocols, mechanisms, or configurations without the need for testing them in real production networks, which could lead to major service disruptions.

Traditionally, the networking community has relied on Queuing Theory (QT) for building network models. However, QT fails to produce accurate estimates in scenarios with realistic traffic models since it imposes strong assumptions on packet arrivals, which typically do not hold in real networks [1]. On the other hand, computational models (e.g., packet-level network simulators) are arguably among the most accurate alternatives to traditional network models. However, such tools are based on simulating individual packet events making them suffer from a high computational cost. This often makes them impractical in scenarios with large traffic volumes and topologies as they can not operate at short time scales [2].

Recently, Machine Learning (ML) has emerged as a new way to model complex systems. In particular, Deep Learning (DL) has proven its capacity of extracting higher-level features from raw data and producing highly accurate models. In the context of computer networks, the main advantage of DL models over other techniques is that they are *data-driven*. Indeed, DL models are trained with real-world data without making any presumption about the system they aim to model. As such, they are able to model complex and non-linear behavior found in networks. Given its ambition, these DL-based models are commonly referred to as *Digital Twins* as they are meant to be physical representations of real physical objects [3, 4].

In the field of network modeling, several DL-based network models works have been already proposed [5, 6, 7, 8]. Existing models take advantage of Graph Neural Networks (GNN) and show outstanding accuracy when modeling network performance metrics, in addition, they are able to generalize to unseen network topologies, configurations, and traffic loads.

However, the main limitation of state-of-the-art models is that they consider the traffic as *Traffic Matrices*. This means that they only consider the aggregated bandwidth (over a certain period of time) between a source and a destination pair. This simplifying assumption works well for certain use-cases, but it is clearly a limiting factor because the most relevant description of network traffic are *flows*.

Networks operate with flows, a flow is an aggregation of packets that share some common characteristics. A common aggregation used in real networks are 5-tuple flows, that is packets that have the same protocol (TCP or UDP), source and destination IP addresses as well as same source and destination ports. Flows are important in networking because applications work with flows, and typically network optimization aims to offer different levels of quality of service to the different flows. Some previous works such as [9] propose working at a flow level to optimize user quality of experience (QoE) for video streaming. Other works like [10, 11] optimizes flow routing in carrier-grade networks. Finally, other works [12] propose a solution to optimize the flow completion in data centers. Several spatio-temporal GNNs have been proposed in the past [13, 14], but they do not target the networking scenario and uniqueness.

Operating at the flow level is challenging, flows are dynamic and have a finite duration that ranges from a few packets (typically ms) to tens of thousands of packets (e.g., backup session) and spans days. In networks, flows are concurrent and at a given instant of time, millions of flows can be active at the same time. As a consequence, modeling flows requires understanding the time dimension and thus, modeling how the state of the network changes over time and as flows are created and destroyed.

In this paper, we present Flow-aware Digital Twin (FlowDT), a new GNN-based model that it is able to understand and model flows. This DT works in the time domain and supports the creation and destruction of flows, as well as flows that dynamically change their characteristics. With this, it is able to provide per-flow metrics (delay and jitter) based on the input flow dynamics as well as network configuration (routing, topology, etc.). We validate the accuracy of the model using a packet-accurate simulator and benchmark its performance against a GRU and a Queuing Theory analytical model.

This work has been supported by the Spanish Government through project TRAINER-A (PID2020-118011GB-C21) with FEDER contribution and the Catalan Institution for Research and Advanced Studies (ICREA).

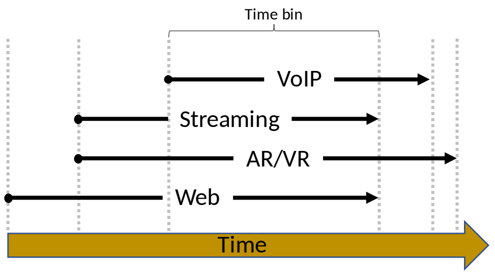

Fig. 1: Representation of the temporal dimension. A change on the network state (e.g, new flow) a new time-bin is considered.

Algorithm 1 Internal architecture of FlowDT

```
Input: $F$, $L$, $x_f$, $x_l$, $h_l^{T(b-1)}$
Output: $h_l^T$, $h_f^T$, $\hat{y_f}$
 1: for each $l \in L$ do $h_l^0 \leftarrow [x_l, 0...0]$
 2: for each $f \in F$ do $h_f^0 \leftarrow [h_l^{T(b-1)}, x_f, 0...0]$
 3: for t = 0 to T-1 do                                  ▷ Message Passing Phase
 4:     for each $f \in F$ do                            ▷ Message Passing on Flows
 5:         for each $l \in f$ do
 6:             $h_f^t \leftarrow FRNN(h_f^t, h_l^t)$            ▷ Flow: Aggr. and Update
 7:             $\widetilde{m}_{f,l}^{t+1} \leftarrow h_f^t$      ▷ Flow: Message Generation
 8:         $h_f^{t+1} \leftarrow h_f^t$
 9:     for each $l \in L$ do                            ▷ Message Passing on Links
10:         $M_l^{t+1} \leftarrow \sum_{f \in L_f(l)} \widetilde{m}_{f,l}^{t+1}$   ▷ Link: Aggregation
11:         $h_l^{t+1} \leftarrow U_l(h_l^t, M_l^{t+1})$      ▷ Link: Update
12:         $\widetilde{m}_l^{t+1} \leftarrow h_l^{t+1}$       ▷ Link: Message Generation
13: $\hat{y_f} \leftarrow R_f(h_f^T)$                          ▷ Readout phase
```

2. NETWORK SCENARIO

We define a flow as an aggregation of packets that have some -loose- common characteristics. Our model supports arbitrary aggregation of packets into flows, for instance, 5-tuple flows. In practice, the main limitation of the aggregation level used with our model depends on the capabilities of the monitoring infrastructure deployed at the network. Operating with fine-grained flows can be computationally expensive [15].

Each flow is defined as a starting time and duration, a source and destination node, and a traffic model. Optionally, it can also include additional information such as ports, protocol, etc. The traffic model describes the inter-arrival time distribution of the packets as well as their packet size. Our model supports arbitrary traffic models, including non-Poisson arrivals with long-tails and auto-correlation.

With respect to the temporal dimension consider figure 1. We define a time-bin as a stationary period for the entire network, this includes all the flows as well as network configuration and topology. Changes in the network state are considered in a new time-bin. Examples of changes in the network state are the creation or destruction of a flow, a flow changing its traffic model or traffic model parameters (increase rate), link failures, or a change in the routing policy. Informally, the DT produces an inference per time-bin, computing per-flow, and per-bin delay and jitter. To produce an estimate, it uses the current state of the network for that time-bin (traffic models, topology, routing, etc.) as well as the previously time-bin state of the network. Also, the DT is suited to work with dynamically changing topologies. This not only means that the model is capable of handling different topologies but also capable of handling topology changes like link failures or network upgrades.

2.1. Use-cases

Digital Twins (DT) are -at the time of this writing- being considered as a key technology for 5G and beyond networks. They offer a fast and accurate representation of the performance of the network and are crucial for optimization, 'what-if' scenarios, and service impact

DT play a role in many different use-cases, in what follows we provide a set of relevant examples. In optimization, optimized configurations need to be tested before being applied, DT can be used to safely test such configurations without disrupting the network. New features, functionalities, and protocols can be tested using a DT, doing this in a production network is very risky. And finally, DTs can play an important role in the training and education of networking experts. The interested reader can find more information about Digital Twins and its use-cases in the following works [4, 3].

3. FLOW-AWARE DIGITAL TWIN

This section describes how FlowDT works, a novel GNN-based solution tailored to accurately model the behavior of real network infrastructures at a flow-granularity level. Particularly, FlowDT describes a new network modeling architecture where the different key elements for network modeling (e.g., forwarding devices, links, flows) exchange messages of their state to the ones they are related with (e.g., via routing).

Specifically, FlowDT (Fig. 2) takes as input (i) a given network configuration (topology, link capacities, and routing) (ii) the per-flow level parameters (iii) the previous bin network state, and produces as output (i) the current network state that will be later used as input for the next time-bin (i) the performance per-flow metrics according to the network state (per-flow mean delay and jitter).

One of the central ideas of FlowDT is that it encodes the state of the network resulting from the previous time-bin, and this state is used to accurately infer the per-flow performance statistics of the current bin. Since a computer network can be understood as a queuing system, we expect this state to be the state of the link/queues.

3.1. Model description

A computer network can be represented by a set of links $L = \{l_i : i \in (1, ..., n_l)\}$, a set of of source-destination flows $F = \{f_i : i \in (1, ..., n_f)\}$, and the routing configuration defined as a set of source-destination path that flows follow. Hence, we define flows as a sequence of the links they traverse $f_i = \{l_{F_l(f_i,0)}, l_{F_l(f_i,1)}, ..., l_{F_l(f_i,|f_i|)})\}$, where the function $F_l(f_i, j)$ return the index of the j-th link along the path of flow f_i.

Using this notation, FlowDT considers three main inputs: (i) the physical links L defined by the network topology, (ii) the active flows F in the network in a specific time-bin, including the source-destination path and the traffic model, and (iii) the network state of the previous time-bin.

The main assumption behind FlowDT is that information at the flow level (e.g., delay) and the link level (e.g., link delay, loss rate, link utilization) can be encoded as vectors of numbers of a given size. Based on this, the main intuition behind this architecture is:

1. The state of flows is affected by the state of the links they traverse.
2. The state of links depends on the states of the flows that go through them.
3. The initial state of the network depends on the previous time-bin network state.

Formally, we define the state of a flow as a hidden vector h_f of a given size. In a similar way, the state of a link is also defined by a

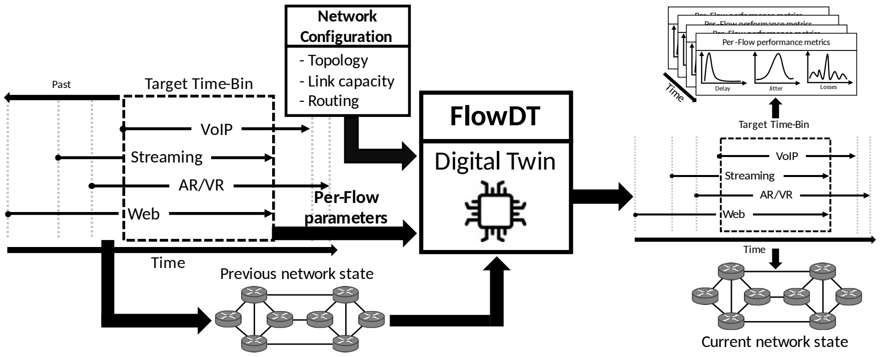

Fig. 2: FlowDT diagram. FlowDT takes as inputs the network configuration (e.g., topology, link capacities, routing), the target per-flow parameters, and the previous network state. FlowDT outputs the per-bin and per-flow performance metrics (delay and jitter) and the current network state.

hidden vector h_l. Knowing this, the principles described above can be formally formulated as follows:

$$h_{f_k} = g_f(h_{l_{f_k(0)}}, h_{l_{k(1)}}, ..., h_{l_{f_k(|f_k|)}}) \quad (1)$$

$$h_{l_j} = g_l(h_{f_1}, ..., h_{f_m}), \quad l_j \in f_k, k = 1, ..., i \quad (2)$$

Where g_f and g_l are some unknown functions that the model needs to learn. Note that a direct approximation of functions g_f and g_l is not possible due to the circular dependencies found between the two components (L and F).

We expect that h_f and h_l encode important information like the per-flow throughput/losses or the per link utilization. However, we do not make any assumptions and let the model learn it directly from the data that is fed into the model.

The architecture of FlowDT (see Algorithm 1) is specifically designed to deal with those circular dependencies via an exchange of messages between links and flows. FlowDT receives as input the initial link and flow features (x_l and x_f respectively), and the network state computed in the previous time-bin ($h_l^{T(b-1)}$). In our particular case x_l is defined as: (i) the link capacity and (ii) the buffer size. On the other side, the initial flow features x_f are the features that describe the behavior of the flow (See section 4.1).

First, in Algorithm 1, the hidden states h_l and h_f are initialized (line 1 - 2) using the initial features described before. In the h_l case, note that the feature vector (x_l) is concatenated with the previous network state ($h_l^{T(b-1)}$). If we are in the first time-bin, the $h_l^{T(b-1)}$ vector is encoded as an array of 0.

Once the initial states are initialized a message-passing phase starts. This message passing phase is executed iteratively during T times (line 3). This process is done to ensure that the state of all the elements has converged. Each message passing iteration can be divided into two stages where the different elements receive information of the elements they are related with, then, this information is aggregated and used to update their state ($FRNN$ and U_l)[1] and finally, create the new message that will be sent. Particularly, the flow aggregation, update and message creation are described in lines 4 - 8, and the link ones in lines 9 - 12.

Finally, the readout phase is executed (line 13). In it, the function R_f is responsible for producing the per-flow level metrics.

4. EXPERIMENTAL EVALUATION

To train, validate and test FlowDT we use as ground truth a packet-level network simulator (OMNeT++ v5.5.1 [17]), where network samples are labeled with performance metrics, including the flows mean delay and jitter over time. In order to generate the dataset, for each sample, we randomly select a combination of input features (flow duration, traffic model, topology, and routing) according to the descriptions below.

4.1. Traffic models

In our experiments, traffic is generated using five different models that range from a Poisson generation process to more realistic (non-Poisson) traffic generation [18]. The different implementation details of these models are defined below (the well-known Poisson and Constant Bitrate are not described since their only configurable parameter is the traffic intensity).

The creation of each flow is based on a Poisson process. The flow duration is based on the distributions described in [19]. All flows have a maximum duration of 1000s which represents 99.5% of all the flows measured in [19]. Note that the flow-creation time and the average traffic per flow have been manually configured to produce low to high congestion levels (to a maximum $\approx 3\%$ of packet loss) similarly to what has been experimentally measured [20].

On-Off: This model is defined by two periods of states (On or Off). The lengths of both periods are randomly selected $[5, 15]$ seconds. During the On period (where packets are transmitted), packets are generated using an exponential distribution.

Autocorrelated exponentials: This model generates autocorrelated exponentially distributed traffic staring from the following auto-regressive (AR) process: $z_{t+1}{=}az_t{+}\varepsilon$, $\varepsilon{\sim}\mathrm{N}(0, \sigma^2)$ where $a{\in}(-1, 1)$ controls the level of autocorrelation. The marginal distribution of z is $\mathrm{N}(0, s^2 = \sigma^2/(1-a^2))$, so z can be negative. In order to construct positive inter-arrival times, z is mapped by a nonlinear transformation: $\delta_t = F_E^{-1}\left(\lambda, F_N\left(0, s^2, z_t\right)\right)$, where $F_N(0, s^2, \cdot)$ and $F_E(\lambda, \cdot)$ are respectively a CDF of the normal distribution with $\mu = 0$ and variance s^2=[3, 12], and an exponential distribution with parameter λ=[40, 2000].

Modulated exponentials: This model is inspired by [21]. It represents an alternative to autocorrelation with higher complexity. Particularly, the inter-arrival times are set by a hierarchical model. In this case, inter-arrivals follow an exponential distribution whose rate is controlled by the value of a hidden AR model similar to the one described before.

4.2. Topologies

To train and test FlowDT, we used three different real-world topologies that have already been used in previous works [7, 22]. Specifi-

[1] $FRNN$ and U_l are implemented as Gated Recurrent Units (GRU) [16], additional details can be found in section 4.4

cally, to train we used NSFNET [23], and GEANT [24] topologies. Then, we validate the accuracy of the model in GBN [25].

4.3. Baselines

To analyze the accuracy of FlowDT, we benchmark it against a stateless Queueing Theory model (s-QT) based on the work described in [7] where the system is modeled as a concatenation of a finite $M/M/1/b$ queues. Note that the s-QT model does not use the state of the network in the previous time-bin. The second benchmark is a Gated Recurrent Unit (GRU) built using an RNN [26]. In this case, the different flows are modeled following a similar strategy of FlowDT where flows are represented as a sequence of links defined by the routing configuration. The main difference between the GRU baseline and FlowDT is that GRU treats each path as a sequence of links not working always with the same states while FlowDT iterates over all the elements of the states of the network updating their states in each one.

4.4. Training and evaluation

We implemented FlowDT as well as the baselines using Tensorflow. In total, FlowDT was trained using 120,000 samples (NSFNET and GEANT) for training and evaluated using 60,000 (GBN) more samples. More information about the topologies used can be found in section 4.2.

In our experiments we use a hidden vector size of 32 for both h_l and h_f. In our particular case x_l is defined as: (i) the link capacity and (ii) the buffer size. On the other side, the initial flow features x_f are defined by the parameters of each distribution described in section 4.1s (on and off times, α, λ), the total number of packets, and the generated traffic. Note that, since we are working with synthetic data the flow features are initialized with the parameters used to create those distributions. However, in a real-world scenario, these features could be learned directly from the data using a specialized module. The total number of iterations (T) is 8.

The functions found in Algorithm 1 are implemented as follows: $FRNN$ (line 6) and U_l (line 11) as Gated Recurrent Units (GRU) [16], and the function R_f (line 13) as a 2-layer fully-connected neural network with ReLU activation functions. It is important to note that the architecture of FlowDT (Algorithm 1) has been specifically designed to be differentiable in order to train the model end to end. Hence, all the different functions that shape its internal architecture are jointly optimized during training based on FlowDT's inputs (network samples) and outputs (per-flow performance metrics).

During the training, we selected as loss function the Mean Squared Error (MSE). This loss function is minimized using an Adam optimizer with an initial learning rate of 10^{-3}.

	Delay				Jitter			
	MAPE	MSE	MAE	R^2	MAPE	MSE	MAE	R^2
s-QT	0.35	0.72	0.43	0.56	0.69	0.29	0.29	0.02
GRU	0.50	1.15	0.54	0.29	1.37	0.16	0.22	0.45
FlowDT	**0.035**	**0.006**	**0.035**	**0.995**	**0.112**	**0.004**	**0.032**	**0.983**

Table 1: Performance comparison for NSFNET and GEANT networks seen during training.

Table 1 shows a summary of the delay and jitter experiments for the GBN topology (never seen during training). We provide 4 metrics: the Mean Absolute Percentage Error (MAPE), the Mean Squared Error (MSE), the Mean Absolute Error (MAE), and the Coefficient of Determination (R^2).

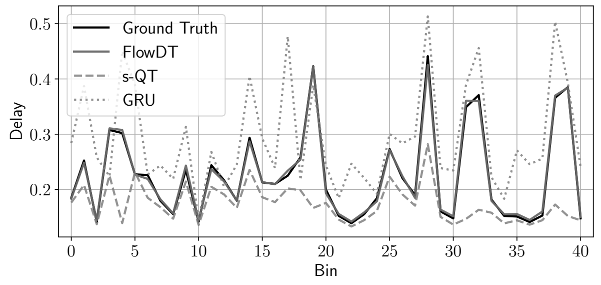

Fig. 3: Delay prediction of one randomly selected flow of the GBN topology.

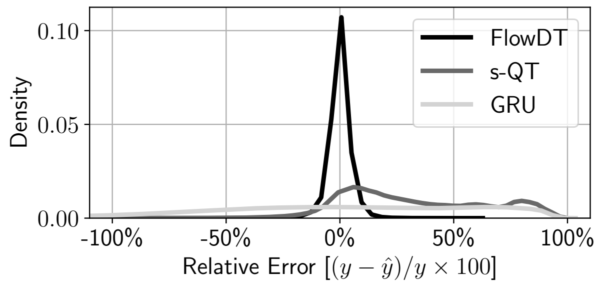

Fig. 4: PDF of the Relative Error reported for the delay prediction.

As can be seen in Table 1, the DT clearly outperforms all the benchmarks. This high accuracy shows how the described DT is able to generalize to unseen topologies, routings, and per-flow characteristics.

Figure 3 shows an example of the time-series for a randomly selected flow. As can be seen, the DT is able to react to the dynamic behavior of the flows, providing accurate estimates of the delay experienced by each flow. In contrast, both baselines seem to provide pretty good estimates when the delay of a time-bin is close to 0.1. However, in peak delays where the network is more saturated, they clearly fail with s-QT underestimating and GRU overestimating the delay. Note that in this experiment only flow-related features are changed over time (e.g., packet rate, distribution, flow parameters). However, the model also accepts changes in the topology (e.g., link capacities, link failures, routing).

In order to analyze how the residuals are distributed we plot them in Figure 4. The figure shows the Probability Density Function (PDF) of the Relative Error of the three models. As the figure shows, the DT produces estimates with the error centered around 0. It looks like, while GRU shows poor performance, the s-QT model undershoots the true values. This is in line with previous experimental results [7], the key insight is that queuing theory assumes markovian arrivals while in practice, flows have highly autocorrelated traffic models which result in bursts of packets that increases the queue weighting time.

We experimentally evaluated the performance of our model with larger topologies than those seen in training, not shown in this paper. Although the error increased significantly with the original training, we observed that only by adding larger link capacities to the training set, the relative error can be reduced to 5%.

5. CONCLUDING REMARKS

In this paper, we have presented FlowDT, a novel Graph Neural Network architecture for network modeling that supports a time-series of flows. FlowDT shows remarkable performance, outperforming state-of-the-art baselines, even in the presence of non-Poissonian flow traffic models.

6. REFERENCES

[1] Zhiyuan Xu et al., "Experience-driven networking: A deep reinforcement learning based approach," in *IEEE INFOCOM*, 2018, pp. 1871–1879.

[2] Shihan Xiao et al., "Deep-q: Traffic-driven qos inference using deep generative network," in *ACM SIGCOMM Workshop on Network Meets AI & ML*, 2018, pp. 67–73.

[3] Liang Zhao, Guangjie Han, Zhuhui Li, and Lei Shu, "Intelligent digital twin-based software-defined vehicular networks," *IEEE Network*, vol. 34, no. 5, pp. 178–184, 2020.

[4] Milan Groshev, Carlos Guimarães, Jorge Martín-Pérez, and Antonio de la Oliva, "Toward intelligent cyber-physical systems: Digital twin meets artificial intelligence," *IEEE Communications Magazine*, vol. 59, no. 8, pp. 14–20, 2021.

[5] Albert Mestres et al., "Understanding the modeling of computer network delays using neural networks," in *ACM SIGCOMM BigDAMA workshop*, 2018, pp. 46–52.

[6] Mowei Wang et al., "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, no. 2, pp. 92–99, 2017.

[7] Krzysztof Rusek, José Suárez-Varela, Paul Almasan, Pere Barlet-Ros, and Albert 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.

[8] José Suárez-Varela, Sergi Carol-Bosch, Krzysztof Rusek, Paul Almasan, Marta Arias, Pere Barlet-Ros, and Albert Cabellos-Aparicio, "Challenging the generalization capabilities of graph neural networks for network modeling," in *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*, 2019, pp. 114–115.

[9] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 197–210.

[10] Renaud Hartert, Stefano Vissicchio, Pierre Schaus, Olivier Bonaventure, Clarence Filsfils, Thomas Telkamp, and Pierre Francois, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," *ACM SIGCOMM computer communication review*, vol. 45, no. 4, pp. 15–28, 2015.

[11] Long Qu, Maurice Khabbaz, and Chadi Assi, "Reliability-aware service chaining in carrier-grade softwarized networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 558–573, 2018.

[12] David Zats, Tathagata Das, Prashanth Mohan, Dhruba Borthakur, and Randy Katz, "Detail: Reducing the flow completion time tail in datacenter networks," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, 2012, pp. 139–150.

[13] Elvin Isufi and Gabriele Mazzola, "Graph-time convolutional neural networks," in *2021 IEEE Data Science and Learning Workshop (DSLW)*. IEEE, 2021, pp. 1–6.

[14] Rongzhou Huang, Chuyin Huang, Yubao Liu, Genan Dai, and Weiyang Kong, "Lsgcn: Long short-term traffic prediction with graph convolutional networks.," in *IJCAI*, 2020, pp. 2355–2361.

[15] Alessandro D'Alconzo, Idilio Drago, Andrea Morichetta, Marco Mellia, and Pedro Casas, "A survey on big data for network traffic monitoring and analysis," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 800–813, 2019.

[16] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[17] András Varga, "Discrete event simulation system," in *European Simulation Multiconference (ESM)*, 2001, pp. 1–7.

[18] J Popoola and RA Ipinyomi, "Empirical performance of weibull self-similar tele-traffic model," *International Journal of Engineering and Applied Sciences*, vol. 4, no. 8, pp. 257389, 2017.

[19] Myung-Sup Kim, Young J Won, and James W Hong, "Characteristic analysis of internet traffic from the perspective of flows," *Computer Communications*, vol. 29, no. 10, pp. 1639–1652, 2006.

[20] Simon Bauer, Benedikt Jaeger, Fabian Helfert, Philippe Barias, and Georg Carle, "On the evolution of internet flow characteristics," in *Proceedings of the Applied Networking Research Workshop*, 2021, pp. 29–35.

[21] Thomas Karagiannis, Mart Molle, Michalis Faloutsos, and Andre Broido, "A nonstationary poisson view of internet traffic," in *IEEE INFOCOM 2004*. IEEE, 2004, vol. 3, pp. 1558–1569.

[22] José Suárez-Varela, Miquel Ferriol-Galmés, Albert López, Paul Almasan, Guillermo Bernárdez, David Pujol-Perich, Krzysztof Rusek, Loïck Bonniot, Christoph Neumann, François Schnitzler, et al., "The graph neural networking challenge: a worldwide competition for education in ai/ml for networks," *ACM SIGCOMM Computer Communication Review*, vol. 51, no. 3, pp. 9–16, 2021.

[23] Xiaojun Hei, Jun Zhang, et al., "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.

[24] Fernando Barreto et al., "Fast emergency paths schema to overcome transient link failures in ospf routing," *arXiv preprint arXiv:1204.2465*, 2012.

[25] João Pedro, João Santos, and João Pires, "Performance evaluation of integrated otn/dwdm networks with single-stage multiplexing of optical channel data units," in *International Conference on Transparent Optical Networks*, 2011, pp. 1–4.

[26] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.