Digital Twin for Networking:
A Data-driven Performance Modeling Perspective

Linbo Hui, Mowei Wang, Liang Zhang, Lu Lu, and Yong Cui

Abstract—**Emerging technologies and applications make the network unprecedentedly complex and heterogeneous, leading physical network practices to be costly and risky. The digital twin network (DTN) can ease these burdens by virtually enabling users to understand how performance changes accordingly with modifications. For this "What-if" performance evaluation, conventional simulation and analytical approaches are inefficient, inaccurate, and inflexible, and we argue that data-driven methods are most promising. In this article, we identify three requirements (fidelity, efficiency, and flexibility) for performance evaluation. Then we present a comparison of selected data-driven methods and investigate their potential trends in data, models, and applications. Although extensive applications have been enabled, there are still significant conflicts between models' capacities to handle diversified inputs and limited data collected from the production network. We further illustrate the opportunities for data collection, model construction, and application prospects. This survey aims to provide a reference for performance evaluation while also facilitating future DTN research.**

Index Terms—**Network Performance Evaluation, Digital Twin Network, Data-driven Performance Modeling**

I. INTRODUCTION

THE endless pursuit for high-throughput and low-latency urges emerging technologies (e.g., 5G, cloud computing, edge computing) to be employed. Meanwhile, high-performance network services, in turn, have spawned a batch of new applications (e.g., live streaming, virtual reality, cloud gaming). These technologies and applications make the network unprecedentedly complex and heterogeneous, leading to costly and risky practices on the physical network. In this context, a digital twin network (DTN) [1] can significantly ease the practitioners' burdens. DTN is a virtual representation of the physical communication network that continuously updates with the latter's performance, maintenance, and health status data. Unlike typical digital twin technologies that replicate physical objects, DTN is primarily concerned with the abstraction of network states and behaviors. DTN aims to build digital twins for universal communication networks and is not restricted to specific applications or contexts.

Network operators often desire to develop optimization techniques to improve network performance, which mainly involves configuration tunning and new-policy exploration. New configurations and policies must be fully verified before being deployed in physical networks. DTN can be used as a safe and cost-efficient environment for performance evaluation. Operators may explore and verify their new techniques

Linbo Hui, Mowei Wang, and Yong Cui are with Tsinghua University (Yong Cui is the corresponding author, e-mail cuiyong@tsinghua.edu.cn); Liang Zhang is with Huawei Technologies Co., Ltd; Lu Lu is with China Mobile Communications Group Co., Ltd.

in DTN, avoiding complicated and risky operations on physical networks. The optimization and other scenarios (§II.A) demand performance evaluation for "What-if" scenarios [2], which means the DTN can tell what the network performance is if there are alterations in influencing factors (e.g., traffic volumes, device configurations, routing schemes, topologies).

Network performance evaluation has attracted researchers' interest for decades. Experiments (e.g., A/B testing) and measurements are two techniques for performance evaluation with physical production networks, which are both high-risk, high-overhead, and high-complexity for "What-if" scenarios. In virtual environments, simulation and modeling are two fundamental approaches for "What-if" performance evaluation. Network simulators (e.g., NS-2, NS-3, OMNet++) process virtual packets under pre-defined mechanisms (e.g., congestion control algorithms, queueing policies) and generate performance metrics (e.g., throughput, delay, loss rate), which allow the collection of arbitrary information without impacting system behavior. Such packet-level simulators are delicately designed and tightly coupled, leading to **inefficient** execution. The 3-4 orders of magnitude slower than real-time [3] determine that current simulators are unacceptable for DTN. Modeling is much different from simulation, which directly establishes the relationships between influencing factors and performance metrics. Conventional analytical modeling methods (e.g., network calculus, queuing theory) adopt Poisson Process to simplify the packet arrival and departure process, which can not describe the incast [4] and leading to **inaccurate** estimation. Simulators are heavy, while analytics are formalistic, and both are **inflexible** towards rapid and continuous network evolution.

With the renaissance of machine learning in recent years, data-driven techniques, especially neural networks (NNs), seem promising for DTN's performance evaluation. Data-driven methods usually pre-define a series of possible mapping functions and utilize abundant data to determine a set of parameters (i.e., training) that accurately map the influencing factors to performance metrics. The mapping function is able to describe various relationships, which provides flexibility for modeling complex network mechanisms. Trained data-driven models are lightweight and can efficiently generate outputs with only one-time forward computation. Researchers have developed specialized structures for specific tasks (e.g., convolutional neural networks for computer vision, recurrent neural networks for natural language processing) and applied techniques (e.g., regularizations, dropout) to solve the over-fitting and generalization problems, making trained models reliable. With these advantages, data-driven methods can help to mitigate the inefficiency, inflexibility, and inaccuracy issues.

In this article, we identify three requirements (fidelity, efficiency, and flexibility) for network performance evaluation by investigating four typical network scenarios. Then we make a comprehensive comparison of selected data-driven performance models on data, models, and applications. The data utilized are diversified and mainly collected from simulation environments. The models develop from the classical-method stage, vanilla-NN stage to the customized-NN stage and strive to meet three requirements. Though enabled extensive applications, models' practical usages are still rare, which indicates the significant conflicts between models' powerful expression abilities and the shortage of practical data from the production environment. Based on the above, we further describe opportunities and challenges for performance evaluation from data collection, model construction, and application prospects. DTN is undergoing rapid development, and performance evaluation is essential for DTN's construction. This article surveys network performance evaluation from a data-driven perspective. We hope that this survey will not only serve as a favorable reference for performance evaluation but also facilitate future research towards DTN.

II. REQUIREMENTS FOR PERFORMANCE MODELS

A. Scenarios and Requirements

Researchers have proposed a general DTN architecture [1] including three layers, as Figure 1 shows. It needs to periodically collect the static data (e.g., topology, configurations) and continuously collect the runtime data (e.g., link utilization, traffic volume) and populate them into DTN. DTN then reconstructs the internal relations of collected data to represent the physical network state, which enables the "What-if" ability. DTN will benefit network practices by providing a real-time and zero-risk performance evaluation environment. We investigate four typical network scenarios of planning, operation, optimization, and upgrade to identify specific requirements for performance models.

- In the planning scenario, designers need to ensure the planning network's overall performance meets the requirements of the given topology, configurations, and demand traffic. The model must generate performance results under different topologies, configurations, and traffic loads, which requires the model to be accurate on various inputs combinations.
- When in operation, engineers hope to know about the real-time performance changes and quickly respond to potential anomalies. These anomalies must be quickly located or detected once they appear. Real-time monitoring and anomaly detection require the model to efficiently depict the physical network performance.
- Network optimization usually involves configurations tuning and new-policy exploration. Configurations and policies must be fully verified before being deployed in practice. The model is an ideal zero-risk environment to explore schemes and evaluate their performances under various scenarios before deployment.
- Network upgrade often includes topology changes and link expansions. Operators may wonder how to change
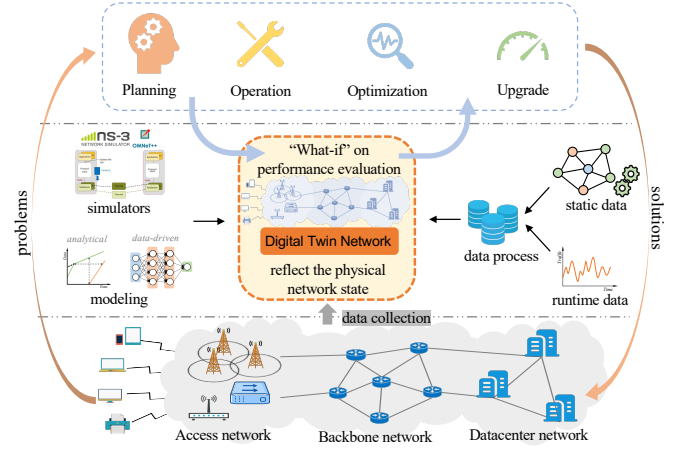

Fig. 1: The Digital Twin Network Architecture.

the topology and where to expand the bandwidth under a limited resource budget. The model needs to evaluate the performance under changed topologies and link capacities and tell where the bottlenecks are to maximize the upgrade effectiveness.

From the above respects, there are three requirements (i.e., fidelity, efficiency, and flexibility) that a performance model must strive to achieve. Fidelity is the basic requirement that ensures accuracy for all scenarios. Efficiency is essential for operation and optimization because of real-time and frequent performance evaluation. Flexibility mainly means the model can evaluate performance with network changes, facilitating network planning, optimization, and upgrade. We further elaborate on three requirements as follows.

B. Requirements Elaboration

1) Fidelity: The fidelity shows how accurate the metrics from performance models are with the physical network. We divide the fidelity into three levels (i.e., long-term, short-term, one-to-one) from the temporal perspective. Long-term represents statistic results over a period of time, while short-term describes the detailed changes in time slots, and one-to-one precisely depicts every packet of the physical network. Long-term **steady** evaluation requires the model to reasonably abstract the complex mechanisms, such as RouteNet [5] can represent arbitrary routing schemes. It further requires describing network conditions and **temporal** dependencies to represent the short-term process, such as xNet [6] learning the state transition function between time steps. One-to-one means we must accurately model diverse mechanisms' influence on each packet. MimicNet [3] keeps end-hosts function and models network clusters' effects on packets, which takes the first step towards one-to-one modeling.

2) Efficiency: Efficiency stands for two parts. One is that performance models can be faster than the real-time physical network. The other is that models should be easy to deploy and consume rational resources. Faster than the physical enables users to forecast performance and react in advance to tackle potential anomalies. Unlike simulators, it needs to simplify

complex mechanisms and discard unnecessary details (e.g., packet payload, switching process) to speed up the evaluation. There is often a trade-off between the modeling granularity (e.g., packet-level [3], flow-level [6], [7], path-level [5], [8]) and speed, which are determined by the target problem settings. The model needs to consider the trade-off and serve as a cost-effective backend to evaluate performance metrics.

3) Flexibility: The network is evolving rapidly. When topology, configurations, or mechanisms change, the model must accurately generate performance metrics as well. We hope that the model can be iteratively upgraded towards new mechanisms, which means the model must be flexible. There usually remain unchanged parts in a changing system, inspiring us to leverage layered or modular philosophies with the flexibility problem. Layers and modules are decoupled but orchestrated to function in one model, where we can separately construct every module. New mechanisms will not affect the whole model but only related modules, which can be either upgraded or replaced. Flexibility will accelerate the process of building an accurate performance model.

III. SELECTED PERFORMANCE MODELS

A. Advances Overview

Network researchers have shown great interest in data-driven performance modeling in the past ten years. We select some representatives and make a comprehensive comparison of problem scopes, data, models, applications in Table I. They are divided into three groups by applied layers. Existing approaches are developed under different performance evaluation tasks where metrics include delay, throughput, loss rate, flow completion time (FCT), etc. Some works focus on performance inference or estimation, and others focus on performance prediction. There are no significant differences between the two focuses, as they just mean the evaluation of current or future performance.

Data, model, and application are three main aspects of a data-driven approach. Data is essential to train the model, which is distinct from conventional analytical or simulation approaches. To some extent, the available data in quantity and quality limits how accurate a data-driven model will perform. Meanwhile, models should be reasonably designed, and a well-designed model can efficiently extract variables' relations to construct itself. Researchers have also developed specific techniques on model architecture and loss function to obtain more accurate results. Most models are not only applied for targeted tasks but also leveraged for other scenarios, demonstrating the broad prospects of performance models.

From an overview, we summarise potential trends in data, models, and applications. The data utilized are diversified but mainly acquired from the simulation environment, indicating the major conflict between models' ability to handle complex inputs and the shortage of data collected from the production network. Diversified data can yield better fidelity and practical applications. Further, the leveraged techniques are advancing with machine learning development, resulting in more delicate models. With the stronger abstraction and powerful expressiveness, models' fidelity and flexibility are improved. At last,
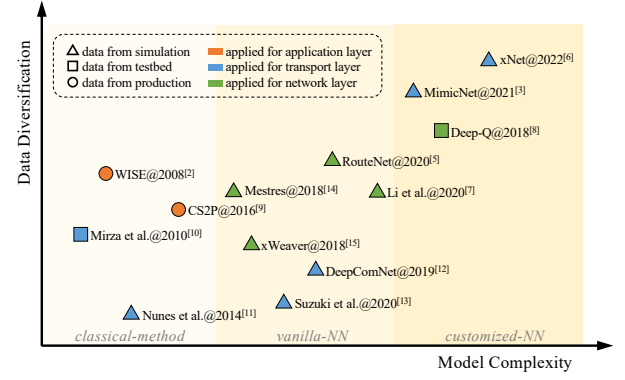

Fig. 2: Qualitative comparison on data diversification, model complexity and application scenarios of selected data-driven performance models

these models have enabled extensive application scenarios for different layers, mainly including network, transport, and application layers. The lower the layer, the more detailed modeling abstractions are required. With the advancement of modeling techniques, there is roughly a tendency from higher to lower layer scenarios. Researchers also realized the great potential of performance models and utilized them to solve problems that seemed to be challenging in the past, such as modeling the routing schemes [5]. Here we give a qualitative comparison of data diversification, model complexity, and application scenarios of selected models as Figure 2 shows.

B. Data Source, Inputs and Outputs

1) Data source: Data source mainly involves simulation, testbeds, and production, where the measurement difficulty sharply increases. We notice that most of the selected works [3], [5]–[7], [11]–[15] are collecting data from the simulation environment. They often leverage packet-level or flow-level simulators, which are costless and relatively easy to deploy. Though various data are convenient to collect in such a restrained environment, the data fidelity may deviate from production. Also, large-scale simulations with packet-level simulators may need a long time to complete.

Some works [8], [10] build testbeds to obtain data. Despite the high cost, testbeds are closer to the real environment and have fast speed. Researchers can collect abundant clean data from controlled testbeds. However, both simulators and testbeds have problems with the traffic model. The flow patterns and distributions are hard to describe because of complex and diverse traffic behaviors. Without reliable traffic models, the credibility of collected data will reduce.

Directly measuring data from production networks is wonderful, where simulation time and traffic model are not problems anymore. WISE [2] used tcpdump data from Google's global web-search content delivery network (CDN), and Sun et al. [9] used a proprietary dataset of HTTP throughput measurement from the operational platform of iQIYI. Real data can bring higher credibility and enables the performance models to be deployed for practical usage. However, collecting plenty of consistent and clean data from the production environment

TABLE I: Comparison of Selected Data-driven Performance Models on Data, Models and Applications

Groups	Authors Year@Pubs	Problem Scopes	Data (Source, Inputs and Outputs)	Models (name, techniques, etc.)	Applications
Application Layer	Tariq, et al. 2008@ SIGCOMM [2]	estimate service response time	**S**: Google's global web-search CDN **I**: tcpdump data with many features (e.g., timestamp, region, RTT, response time) **O**: service response time distribution	WISE uses Causal Baysian Network to learn the causal structure and applies statistical intervention to predict the response time	evaluate the response time under "What-If" scenarios
	Sun, et al. 2016@ SIGCOMM [9]	predict throughput	**S**:iQIYI's operational CDN platform **I**: clientIP, ISP, AS, city and server **O**: throughput	CS2P learns the parameters of Hidden Markov Model (HMM) via expectation-maximization (EM) algorithm	predict throughput for video bitrate adaptation
Tranport Layer	Mirza, et al. 2010@ToN [10]	predict TCP throughput	**S**: laboratory WAN testbed **I**: transfer size and path properties (e.g., queuing delays, loss, available bandwidth) **O**: TCP throughput	Support Vector Regression (SVR) with Radial Basis as kernel function, loss funciton is ϵ-insensitive loss with L2 regularization	predict end-to-end TCP throughput of wide area paths
	Nunes, et al. 2014@JWCN [11]	estimate RTT of TCP connection	**S**: QualNet simulation **I**: three hyperparameters, measured RTT **O**: next RTT	online fixed-share experts learning with weights updated every trail, loss is a piecewise function	implemented inLinux kernel to estimate TCP RTT
	Geyer, 2019@ Performance Evaluation [12]	evaluate the performance of a topology	**S**: from simulation **I**: graph with nodes of flows and queues **O**: throughput of TCP flows, end-to-end latencies of UDP flows	DeepComNet uses Graph Neural Network (GNN) with Gated Recurrent Units, loss function is MSE	predict average TCP flows bandwidths and UDP flows end-to-end latencies
	Suzuki, et al. 2020@ICOIN [13]	infer end-to-end delay	**S**: Fluid-based Simulator **I**: some delays of node-pairs and node features (e.g., indicator and degree) **O**: delays of other node-pairs	semi-supervised GCN learning with rectified linear unit andlogarithmic softmax classifier, loss function is negative log likelihood	infer delays of other node pairs from measured delays at some nodes.
	Zhang, et al. 2021@ SIGCOMM [3]	model DCN clusters' effects on packets.	**S**: OMNet++ simulation **I**: scalable features (e.g., number of racks per cluster, packet size, priority bits) **O**: packet latency in the cluster	MimicNet's internal model uses LSTM model, loss fucntion is Weighted-BCE with Huber, two losses are weighted	scalable, faster, and tunable performance estimation for DCN
	Wang, et al. 2022@ INFOCOM [6]	model network performance	**S**: NS-3 simulation **I**: traffic, buffer size, ECN, topology, queue policy, routing scheme, etc. **O**: path-/flow-level delay/throughput, FCT	xNet uses three NGN blocks to build the state transition model, L2 loss function is used	online QoS monitoring, "What-if" simulation, offline planning
Network Layer	Xiao, et al, 2018@NetAI [8]	infer path delay and loss distribution	**S**: testbeds of DCN and WAN **I**: ports' load matrix in every four seconds **O**: path delay and loss distribution in every minute	Deep-Q uses a variational auto-encoder (VAE) enhanced by the long short term memory (LSTM), loss function is Cinfer-loss	infer QoS metrics of given traffic matrix
	Mestres, et al. 2018@ BigDAMA [14]	model end-to-end delay	**S**: OMNet++ simulation **I**: traffic matrix of various scenarios **O**: end-to-end delay matrix	neural networks model with different parameters, loss function is MSE with L2 regularization	model end-to-end delay as function of traffic matrix
	Wang, et al. 2018@ SIGMETRICS [15]	evaluate the performance of DCN topology	**S**:customized flow-level simulator **I**: demand traffic matrix and topology configuration **O**: performance metrics score	xWeaver's scoring module uses two convolutional neural networkwith a fully-connected neural network, loss function is not mentioned	evaluate and infer the propertopology of given traffic
	Rusek, et al. 2020@JSAC [5]	model the path delay, jitter or loss	**S**: OMNet++ simulation **I**: topology, routing schemes, link capacities, and path steady traffic **O**: path mean delay, jitter and loss	RouteNet uses Message Passing Neural Network (MPNN) framework, loss is negative log likelihood	QoS-aware routing optimization, budget-constrained network upgrade
	Li, et al. 2020@CN [7]	model FCT in DCN	**S**: DiffservNetwork simulator **I**: flow size and start time, ToS, protocol, bandwidth, etc. **O**: FCT	GNN model (three Graph Networks blocks as encoder, core, decoder), loss function is MSE	flow routing and scheduling, topology management for trafficoptimization

is still costly. Specialized equipment and techniques are often needed to take measurements. Even so, some metrics (e.g., queue length [4]) can not be directly measured. High-precision time synchronization on devices also hinders accurate measurements of time-related metrics.

2) Inputs: Inputs are various for different tasks. Some works [2], [5], [7], [9], [10] take multiple discrete or continuous variables as inputs. WISE [2] inspected a tcpdump dataset and found that a series of variables (e.g., timestamp, number of sent packets, client region, round-trip time (RTT), response time) are responsible for service response time. Mirza et al. [10] proposed that incorporating path properties (e.g., queuing delays, loss, available bandwidth) as inputs could improve history-based TCP throughput prediction. CS2P [9] picked a given set of features from all possible features (e.g., client IP, ISP, AS, city, server) combinations to aggregate session clusters. RouteNet [5] utilized path steady traffic under various topologies, routing schemes, and link capacities to infer the path delay and loss rate. Li et al. [7] took flow size, start time, type of service (ToS), protocol, and link bandwidth

to model flow completion time (FCT). Though it introduces a heavier burden, diversified inputs will yield more accurate outputs and stronger generalization ability.

There are also works [8], [11]–[15] taking fewer features as inputs. Such works often focus on given scenarios and tackle specific tasks. Nunes et al. [11] and Suzuki et al. [13] only used measured metrics (RTT/delays to some nodes) to estimate homogeneous metrics (next RTT/delays to other nodes). xWeaver [15], Deep-Q [8], and Mestres et al. [14] preferred to use traffic matrixes to model end-to-end performance matrixes, where topology and other configurations were specified. DeepComNet [12] employed lower-level features (graph representation of topology and flows) to evaluate the average performance of network topology. Fewer input features can reduce the measurement costs and enable fast execution, improving efficiency under specific settings.

3) Outputs: Outputs are performance metrics (e.g., RTT, delay, throughput, loss rate, FCT) with fewer dimensions than inputs. The majority of the works [5], [7], [9], [10], [12]–[15] aim to evaluate long-term average performance under steady-

state. Mirza et al. [10] and CS2P [9] both predicted throughput in a period. xWeaver [15] and DeepComNet [12] both estimated the given topology's steady performance. Mestres et al. [14] and RouteNet [5] both evaluated path mean delay, which reflected the overall path performance under steady traffic. Suzuki et al. [13] inferred end-to-end delays of node pairs under persistent TCP flows. Li et al. [7] modeld FCT in DCN. Above long-term performance evaluation facilitated decision-making for target tasks.

Unlike the average metrics above, generating metrics distributions enables users to evaluate the performance from a probabilistic point. WISE [2] estimated service response time distribution under "What-if" scenarios and could provide the percentage guarantee for service level agreement (SLA) requirements. Deep-Q [8] verified that some quality of service (QoS) metrics are not a scalar but a random variable. It inferred path delay and loss distributions over time intervals. Deep-Q also provided performance changes over time, which enabled dynamic performance evaluation from a temporal dimension. Nunes et al. [11] estimated the next RTT with past measured RTT. xNet [6] enabled temporal prediction on QoS inference and FCT. MimicNet [3] modeled latency on each packet. The temporal metrics reflect performance changes over time, enabling more valuable applications (e.g., QoS monitoring, anomalies detection, exploring policies).

C. Model Selection and Customization

From an overview of model construction, we divide recent advances into three stages: classical-method stage, vanilla-NN stage, and customized-NN stage.

1) classical-method stage: In this stage, classical methods (e.g., Causal Bayesian Network (CBN), Support Vector Regression (SVR)) are adopted for performance modeling tasks. Though with fixed and straightforward forms, such methods have shown good performance for specific tasks. WISE [2] identified relevant features from vast variables and leveraged CBN to construct the causal structure of these features. Then it applied statistical interventions to changed features to estimate response time under "What-if" scenarios. Mirza et al. [10] leveraged SVR with Radial Basis kernel function to predict TCP throughput. The loss function is ϵ-insensitive loss with L2 regularization. SVR has a solid theoretical foundation and is favored in practice for its excellent empirical performance. Nunes et al. [11] used lightweight Experts Framework to perform online learning, which showed high speed with reasonable accuracy. They developed a dedicated piecewise loss function to describe the error between prediction and measurement. CS2P [9] employed Hidden Markov Model (HMM) to capture the state-transition behaviors for similar clusters to predict throughout for video bitrate adaptation.

In this stage, classical data-driven techniques often have fixed forms and few empirical hyperparameters, where loss functions are easy to determine. Based on this, there is not much space to design particular architectures for specific tasks. Researchers need to abstract the problems and pay attention to feature engineering. These models are very efficient and have high fidelity when properly applied, while they may not be flexible for network evolution. With fixed forms, it is challenging for them to model complicated network mechanisms, leading to very limited applications.

2) vanilla-NN stage: With the development of computing technology and abundant available data, NNs bring data-driven techniques into a new era. NNs are adopted for network performance modeling as well, and this enables the vanilla-NN stage. xWeaver [15] and DeepComNet [12] both leveraged NN techniques to evaluate the performance of a given topology. xWeaver can explore optimal topology for given traffic, where a scoring module was used for evaluation. The scoring module used two Convolutional Neural Networks (CNNs) to extract information from traffic and topology configuration separately, with a fully-connected NN to output the performance score. DeepComNet represented flows and queues as graph nodes. When a flow traverses a queue, an edge is connected with the flow node and the queue node. It then leveraged Gated Graph Neural Network (GGNN) to evaluate topology. Mestres et al. [14] wondered whether NN could accurately model the delay as a function of the input traffic. They evaluated NN with different hyperparameters (i.e., number of hidden layers, number of neurons per layer, the activation function, the learning rate, and the regularization parameter) and gave an affirmative answer. Suzuki et al. [13] performed semi-supervised learning with Graph Convolutional Networks (GCN) to infer end-to-end delay. They modeled the problem as a classification question and used a logarithmic softmax classifier. RouteNet [5] utilized Message Passing Neural Network (MPNN) to model the relationships between links and paths, outputting the path mean delay, jitter, and loss. Li et al. [7] abstracted flows as graph nodes and links as graph edges. They input features of flows and links to the model, built with three Graph Network (GN) blocks, to output FCT.

The above works addressed problems with suitable NN techniques, where model structures and loss functions keep conventional. Emerging NN methods are directly adopted for specific tasks. Their inputs are usually structured (e.g., graph structure) and can not be directly processed by classical techniques. These inputs have multiple dimensions, enabling better flexibility for long-term performance modeling.

3) customized-NN stage: Researchers have realized that detailed performance modeling (e.g., short-time and one-to-one) is more valuable and practical. Deep-Q [8] first modeled path delay and loss distribution at time slots. They combined a variational auto-encoder (VAE) with the long short-term memory (LSTM). LSTM was used to extract information from sequences of traffic matrix, and VAE could generate metric distributions. A specially-designed Cinfer-loss module could measure the error of predicted QoS distributions, which enabled efficient and accurate training. MimicNet [3] provided over two orders of magnitude speedup compared to regular simulation for DCN of thousands of servers. It used a data-driven model to replace the complex and slow clusters in DCN while keeping the scalability, flexibility, and accuracy. For accurately modeling effects (drop, latency, ECN, etc.) on packets, they developed a loss function with Weighted-Binary Cross-Entropy (BCE) and Huber loss. xNet [6] provides a general approach to model the network characteristics of concern

with graph representations and configurable GNN blocks. It learns the state transition between time steps and rolls it out to obtain the entire fine-grained prediction trajectory. xNet used three Networking Graph Networks (NGNs) to build the state transition model and applied loss for state transitions.

These works modeled the network at a dynamic view and provided higher fidelity, where inputs and outputs changed over time. They designed delicate NN models and adopted domain knowledge for accurate modeling.

D. Applications Scenarios

In the beginning, there were not many applications for data-driven models. Many works [2], [9]–[11], [14], [15] focused on single-metric modeling. These works were designed for specific tasks and were only used in the targeted scenarios. WISE [2] evaluated the service response time under the "What-if" scenarios. Nunes et al. [11] used online learning to estimate the RTT of a TCP connection. It could be implemented in the Linux kernel to improve the accuracy of RTT estimation. Mirza et al. [10] and CS2P [9] both predicted throughput under specific background for file transfer and video bitrate adaptation separately. Mestres et al. [14] verified that NN could accurately model end-to-end delay as a function of the traffic matrix. They did not explore the use cases for applications. xWeaver [15] was dedicated to evaluating the performance of topology. It could be used to infer the proper topology for given traffic. Above all, we have seen very limited applications for these works. There may be multiple reasons behind in aspects of data collection and modeling techniques. Fewer inputs, single outputs, and specialized model is only valid for specific tasks. Researchers did not consider applying these models for more scenarios.

With the advances of various NN techniques, models are becoming more powerful. Extensive applications have been proposed for related works [3], [5]–[8], [12], [13]. Deep-Q [8] hoped to infer performance metrics directly from traffic statistics in real-time. It could also be used for performance optimization. DeepComNet [12] was designed to evaluate the performance of topology. Engineers could leverage DeepComNet to predict TCP bandwidth or UDP end-to-end latency for network planning. RouteNet [5] set an excellent example in applying the model for multiple use cases. They modeled path mean delay, jitter, and loss, which are used for QoS-aware routing and budget-constrained network upgrades. Li et al. [7] also showed multiple applications for traffic optimization of flow routing and scheduling, topology management. MimicNet [3] enhanced the packet-level simulator and provided scalable, faster, and tunable performance evaluation for DCN. xNet [6] proposed a modeling framework and demonstrated three use cases of online QoS monitoring, simulation of "What-if" scenarios, and network planning. It is evident that data-driven models have more extensive applications than ever before. Generalized data-driven methods have reduced the difficulties of traditional problems with accurate, flexible, and efficient modeling techniques.

IV. CHALLENGES AND OPPORTUNITIES

Data-driven performance evaluation is rapidly advancing, and multiple applications will benefit from it. Despite the bright prospects, there are still many obstacles to conquer, with opportunities lying behind. We conclude challenges and opportunities with respect to data collection, model construction, and application prospects as follows.

A. Data Collection

Available data for training strongly affects the accuracy and generalization of learning models. Data from production networks often with higher value but the amount and types are limited. Real data for training will introduce a practical model for deployment, such as WISE [2]. At the same time, diverse data need to be collected under various configurations, which is unrealistic in the production network. Simulation environments may be a good alternative to solve the problem of accuracy and diversification. However, such simulations are often time-consuming. In addition, there is often a gap between simulated data and real data, hindering trained models from applying to production. Data-driven techniques have been widely developed in broad fields, where general datasets and benchmarks contribute a lot, while there are still no widely recognized datasets and benchmarks for networking.

Experiments, measurements and data-driven methods are all promising for these problems. Engineers may need to utilize specialized techniques (e.g., high-precision time synchronization, in-network telemetry) and equipment (e.g., sketch-related) to accurately measure valuable data in production [4]. Meanwhile, various data-driven techniques are helping to mitigate the problem too. Network domain knowledge can be imposed on a few measured data for augmentation. Other learning techniques with fewer data are also proposed, such as few-shot learning and self-supervised learning. Transfer learning may help to transform the simulation-data trained model into a practical model with little real data. Data-driven networking is evolving rapidly, and we are confident that standard **datasets and benchmarks** will be formulated and bring significant promotion to this exciting field.

B. Model Construction

Data-driven models need to reflect the complex mechanisms of the network. There are both global and local, spatial and temporal relations of network entities, and they tangle together. Congestion control algorithms, queueing policies, routing schemes, etc., all have essential impacts on network performance. Congestion control algorithms function at flow-level but global, where both end-hosts and in-network information are utilized. Queuing policies schedule packets in the local switching node, influencing the end-hosts behaviors. Routing schemes manage flows at the path level, which has significant effects on traffic distribution. In addition, these mechanisms are time-sensitive (i.e., current state changes will impact future states). How to accurately model the global and local, spatial and temporal mechanisms are still challenging. We also notice that data-driven models often consume many resources on data

collection, training, tuning, etc. It is not trivial to ultimately construct such a model from the blank.

We advocate using the modular or layered conception, and analytical-NN combined cogitation to ease the complex modeling problem. The tangled mechanisms can be decoupled using modular or layered parts that may be built and updated independently. Analytical methods can also be introduced to enable higher accuracy and efficiency, which are often derived from domain knowledge. Current data-driven models are developed separately, while a **foundation model** might be quite beneficial. Pre-trained foundation models can serve as a backend and we only need to fine-tune it for specific tasks. The foundation model for performance evaluation is similar to Transformer for natural language processing, which can provide universal and basic modeling ability. The challenges for constructing foundation models are the complicated model constructions and the universal abstraction of network mechanisms. A potential method is to describe the network as packets sequences, where various mechanisms can be viewed as impactions on the interval time between packets. We are yearning for all kinds of foundation models to appear in the near future.

C. Applications Prospects

Data-driven methods have advantages over conventional methods in efficiency, fidelity, and flexibility, which conducts great potential for the optimization of network configuration. Search-based optimization policies will be promoted with faster evaluation speed. What's more, temporal NN models can efficiently make sequential evaluations, thus enabling model-based control optimizations. With the model faithfully evaluating networks' performance, the DTN can provide a high-fidelity dynamic environment. Reinforcement learning (RL) techniques are often employed for network decision-making. The RL agent needs to perceive states from the environment and make specific actions to maximize rewards. The virtual dynamic DTN environment is an ideal playground for RL algorithms to safely explore new policies. There are also emerging technologies (e.g., Artificial Intelligence for IT Operations (AIOps), Self-driving Networks) to alleviate engineers from heavy manual operation works, where DTN can serve as an exploration and verification environment.

V. Conclusion

Though many challenges ahead, we will ultimately achieve the DTN technology with continuous research, and we are currently taking the first step. The DTN's essential feature is the "What-if" ability, and the performance modeling plays a critical role. Conventional simulation and analytical approaches are inefficient, inaccurate, and inflexible, and we argue that data-driven methods are the most promising to build the performance model. Researchers have proposed a few methods of data-driven performance evaluation, while systematic summaries are still missing. This article surveys selected data-driven performance models from data, models, and applications perspectives. The data utilized are mainly collected from simulation environments, and the models develop from the classical-method stage, vanilla-NN stage to customized-NN stage. Though enabled extensive applications, models' practical usages are still rare, which indicates the significant conflicts between models' powerful expression abilities and the shortage of practical data from the production environment. We believe that standard datasets and benchmarks will be formulated, and foundation models will appear to promote this exciting field. We anticipate that this survey will not only serve as a favorable reference for performance evaluation but also facilitate future research towards DTN.

References

[1] C. Zhou, H. Yang, X. Duan, D. Lopez, A. Pastor, Q. Wu, M. Boucadair, and C. Jacquenet, "Digital Twin Network: Concepts and Reference Architecture," Internet Engineering Task Force, Internet-Draft draft-zhou-nmrg-digitaltwin-network-concepts-07, Mar. 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-zhou-nmrg-digitaltwin-network-concepts-07

[2] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar, "Answering what-if deployment and configuration questions with wise," in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, 2008, pp. 99–110.

[3] Q. Zhang, K. K. Ng, C. Kazer, S. Yan, J. Sedoc, and V. Liu, "Mimicnet: fast performance estimates for data center networks with machine learning," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 287–304.

[4] Y. Geng, S. Liu, Z. Yin, A. Naik, B. Prabhakar, M. Rosenblum, and A. Vahdat, "{SIMON}: A simple and scalable method for sensing, inference and measurement in data center networks," in *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, 2019, pp. 549–564.

[5] 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.

[6] 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 - IEEE Conference on Computer Communications (INFOCOM 2022)*, London, United Kingdom (Great Britain), May 2022.

[7] J. Li, P. Sun, and Y. Hu, "Traffic modeling and optimization in datacenters with graph neural network," *Computer Networks*, vol. 181, p. 107528, 2020.

[8] 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, pp. 67–73.

[9] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 272–285.

[10] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to tcp throughput prediction," *IEEE/ACM Transactions on Networking*, vol. 18, no. 4, pp. 1026–1039, 2010.

[11] B. A. A. Nunes, K. Veenstra, W. Ballenthin, S. Lukin, and K. Obraczka, "A machine learning framework for tcp round-trip time estimation," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, pp. 1–22, 2014.

[12] F. Geyer, "Deepcomnet: Performance evaluation of network topologies using graph-based deep learning," *Performance Evaluation*, vol. 130, pp. 1–16, 2019.

[13] T. Suzuki, Y. Yasuda, R. Nakamura, and H. Ohsaki, "On estimating communication delays using graph convolutional networks with semi-supervised learning," in *2020 International Conference on Information Networking (ICOIN)*. IEEE, 2020, pp. 481–486.

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

[15] M. Wang, Y. Cui, S. Xiao, X. Wang, D. Yang, K. Chen, and J. Zhu, "Neural network meets dcn: Traffic-driven topology adaptation with deep learning," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 2, pp. 1–25, 2018.

Linbo Hui received the B.E. degree from North China Electric Power University, Beijing, China, in 2016. He is currently studying toward his M.E. degree in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His major research interest is machine learning for network modeling.

Mowei Wang received the B.E. degree in communication engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2017. He is currently working toward his Ph.D. degree in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests are in the areas of data center networks and machine learning.

Liang Zhang received the Ph.D. degree from Southeast University, Nanjing, China, in 2010. He is currently Vice-Director of Huawei AI4NET Lab and Director of Huawei DataCom AI Department. His research interests include intelligent fault analysis, network traffic analysis and network optimization.

Lu Lu is currently the deputy director of infrastructural network technology department in China Mobile Research Institute and the vice chairman of ITU SG13 WP1. Her research interest covers mobile core network, future network architecture, computing and network convergence etc.

Yong Cui received the B.E. and Ph.D. degrees both in CS Department from Tsinghua University. He is currently a professor. He served or serves at the editorial boards on IEEE TPDS, IEEE TCC, IEEE Network and IEEE Internet Computing. He co-chaired ACM Sigcomm'19 in Beijing and an IETF working group. He published over 100 papers with several Best Paper Awards and a dozen of Internet standard documents (RFC). His research interests include the Internet architecture and the data-driven network