Towards Intent Assurance: A Traffic Prediction Model for Software-Defined Networks

M. Gharbaoui*, B. Martini†, D. Berardi †, P. Castoldi*
*Scuola Superiore Sant'Anna, Italy; †Universitas Mercatorum, Italy;

2025 28th Conference on Innovation in Clouds, Internet and Networks (ICIN) | 979-8-3315-4239-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICIN64016.2025.10942926

***Abstract*—In modern networks, the increasing complexity and dynamic traffic patterns pose significant challenges to maintaining optimal performance and detecting traffic anomalies that can degrade network performance and lead to violations of Service Level Agreements (SLAs). This paper presents a traffic prediction model designed to prevent congestion in Software-Defined Networks (SDN) as part of the assurance operations within an Intent-Based Networking (IBN) framework. The proposed model leverages real-time network data and machine learning techniques to forecast traffic patterns, enabling proactive congestion management and resource optimization. By predicting potential bottlenecks, the model facilitates dynamic adjustments through load balancing to ensure consistent Quality of Service (QoS) and adherence to network intents. Integrated into the IBN framework, the model enhances both network reliability and user satisfaction.**

***Index Terms*—IBN;Prediction; SDN; Machine Learning; SVM.**

I. Introduction

Intent-Based Networking (IBN) [1] comes into play as a novel approach in network management to (i) enable a loose-lycoupled interworking between applications (i.e., vertical applications or service management applications) and network operators, and (ii) further foster the development of third-party applications thanks to abstractions and semantics not generally available at the northbound interfaces of network control or management systems, e.g., SDN controllers. Indeed, application developers are not inclined to specify low-level technical parameters (e.g., Virtual Local Area Network (VLAN) tags, forwarding rules, connection points) needed to realize their business and operational goals in a specific application domain [2]. In response, the network can autonomously translate intents into appropriate actionable policies while optimizing network configuration and behavior accordingly.

A key aspect of intent-based networking is ensuring that the network status continuously and consistently adheres to the deployed intents and performance goals that align with user expectations. Traffic anomalies or network congestion can degrade network performance, leading to violations of the intents deployed within the network systems. Hence, ensuring the continuous fulfillment of intents is crucial, particularly in dynamic network environments featured by unpredictable traffic patterns and potential congestion [3]. This motivates the need for proactive mechanisms capable of predicting traffic patterns and potential network congestions before they impact performance. Such mechanisms can detect early warning signs and allow for preemptive adjustments to the network configurations in order to keep the network remains aligned with the desired performance objectives [4].

This work presents a predictive traffic model incorporated into an intent layer capable of forecasting potential congestion scenarios. The model is built upon a machine learning approach, specifically using Support Vector Machines (SVM), which can detect early warning signs of network congestion and enable preemptive adjustments to the network or switch configuration. This ensures that the network continues to meet the expected performance goals while minimizing the risk of deviating from expected user intents. Additionally, predictive capabilities can support proactive decision-making processes, enabling the network to guarantee service continuity without manual intervention (i.e., zero-touch), thus ensuring seamless user experiences even in variable traffic conditions.

II. Related Work

Traffic prediction plays an important role in modern network management, particularly within the context of IBN, where the network must dynamically adapt to high-level intents specified by operators. Traditional approaches to traffic prediction typically rely on time series models such as Autoregressive Integrated Moving Average (ARIMA) and its variants [5] or machine learning models to forecast traffic patterns [6][7]. More specifically, in [6], Abbas et. al developed an AI-driven network data analytics function (NWDAF) for proactive and intelligent resource assurance. The NWDAF uses a hybrid stacking ensemble learning (STEL) system to forecast Virtual Network Functions (VNFs) workload and a Machine Learning (ML) technique to perform network anomalies detection. The results demonstrate adequate performance with higher accuracy than existing models and lower error rates. In [7], an IBN system for the orchestration of OpenStack-based clouds and overlay networks between multiple clouds is presented. It leverages on several ML models (e.g., Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRUs) and Support Vector Regression (SVR)) to find the best path for communication between any two clouds. The network prediction in this case helps in better managing and planning network resources during the fulfillment phase while in this work we focus on the assurance operations. Recent studies have also explored the integration of deep learning techniques, such as Graph Neural Networks (GNNs) and attention-based models, to better handle the complex and heterogeneous nature of network traffic [8][9]. These approaches aim to capture both spatial and tempo-

ral correlations across the network, providing more accurate predictions in scenarios where traffic patterns are influenced by multiple factors, such as topology, routing policies, and varying service demands. However, in the IBN context, there is limited work specifically focused on integrating real-time traffic prediction into assurance processes. In fact, the existing literature on assurance in IBN mainly revolves around network verification, monitoring, and compliance checking, with few addressing the predictive aspect of traffic management [1]. This work bridges the gap by proposing a traffic prediction model tailored for assurance in IBN thus enabling proactive decision-making and better adherence to intent-based policies.

III. System Architecture and Design

A. Overview of the Proposed Architecture

In this section, we present an intent framework for the management of SDN-based networks, aimed at providing adaptive service data paths for applications without the need for prescriptive directives. Fig. 1 outlines the framework's functional design, which extends the SDN Network Layer with an Intent Layer that automates resource management (e.g., data forwarding, load balancing). This allows flexible and efficient network path configuration while meeting the application's requirements [10].

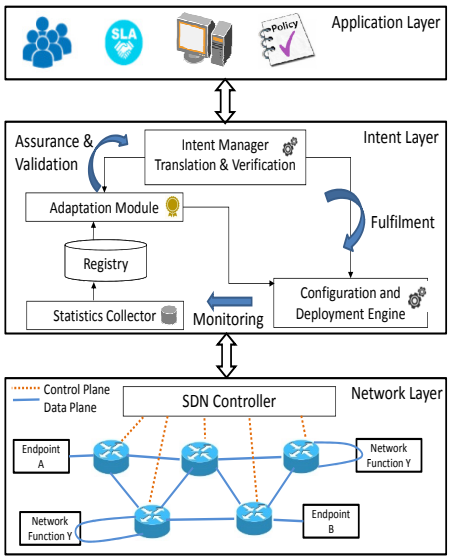

Fig. 1. Intent Framework: Functional design and Building blocks.

The Intent Layer receives service requests from an Application Layer, which specifies high-level goals (i.e., intents) rather than technical constraints. The Intent Manager parses these intents, verifies their feasibility, and ensures they do not negatively impact current network operations. Once validated, the Configuration and Deployment Engine sets up the required service paths across the network and adapts them as needed. The Adaptation Module continuously monitors network performance and triggers re-optimization if the system deviates from intent requirements. Supporting components like the Statistics Collector and Registry provide real-time data to ensure informed decision-making and path adjustments, preserving service quality despite changing network conditions.

Regarding the overall operation of the Intent Layer, it mainly involves three key phases [11]:

- Awareness: The Statistic Collector gathers network statistics via the SDN Controller to assess the network's current status. Switch load is used as a key metric, as it directly impacts throughput and Quality of Experience (QoE). Overloaded switches can increase packet loss and delay, leading to degraded performance.
- Fulfillment & Verification: The Intent Manager translates user goals into network configurations and verifies whether the network can support the requested intent based on resource availability. If successful, the intent is enforced in the network.
- Assurance & Validation: The Intent Layer continuously checks if intents are still meeting the original user goals. The Adaptation Module monitors network performance metrics (e.g., switch load, throughput) and triggers reconfiguration if needed. Actions like path redirection or load balancing are initiated when thresholds, such as switch overload, are exceeded to maintain service performance.

In this paper, we focus on the Assurance & Validation phase where we apply a traffic prediction model to efficiently and dynamically reply to the network changes while maintaining the compliance between the intents goals and the network performance.

B. Traffic Prediction Module

Incorporating a traffic prediction module into the assurance operations of an IBN framework aims to enhance network reliability, efficiency, and user experience by proactively managing traffic flows. More specifically, it allows to predict traffic trends in advance to perform proactive traffic management (e.g., resource allocation, rerouting, and load balancing) and then avoid network congestion. To do so, the module continuously gathers network metrics (e.g., bandwidth, latency, packet flow) to ensure accurate and up-to-date predictions and uses advanced algorithms to analyze historical data and predict future traffic demands. Once done, it interacts with the Intent Manager to ensure that predicted traffic changes are addressed by adjusting the network configuration to meet intent requirements. For example, if the prediction indicates potential performance issues, the module triggers reconfigurations to maintain network performance, guaranteeing that intent-based goals are continuously met, even in dynamic conditions. In this way, the assurance operations are strengthened by enabling the network to automatically adapt to future demands, ensuring high availability, Quality of Service (QoS), and Service Level Agreement (SLA) compliance.

C. Workflow

With reference to Fig. 2, the traffic prediction module follows a structured workflow, which operates by continuously monitoring network data to anticipate future traffic patterns and optimize resource allocation. More specifically, it first

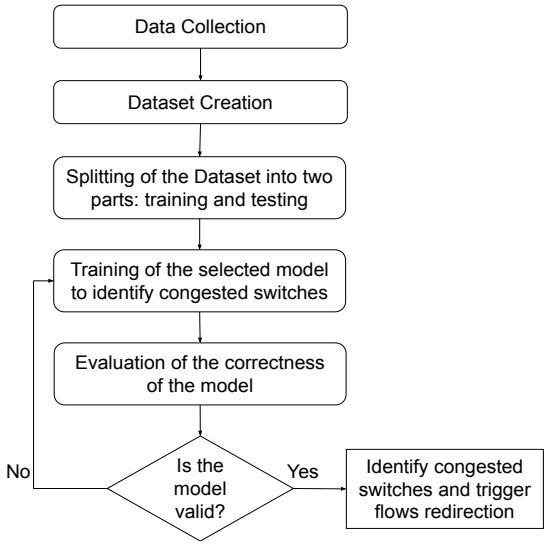

Fig. 2. Traffic prediction workflow.

starts with data collection, where network metrics such as traffic volume, bandwidth usage, latency, and packet flows are gathered from various sources (e.g., routers, switches, sensors). This raw data is processed and used to create a dataset, which aggregates and organizes the information in a suitable format for analysis, typically containing both historical traffic data and relevant features like time, location, and network conditions. Next, the dataset is divided into 2 specific datasets: one for the training used to train the model, usually employing machine learning techniques such as time series analysis, deep learning, or statistical models, and one for testing and validating the model. In fact, the model learns to identify patterns and predict future traffic behavior based on past data. After training, the model undergoes validation, where its predictions are compared against actual traffic data from a separate test dataset to assess its accuracy and generalization ability. If the model performs well, it is deployed to make real-time traffic predictions and identify potential congestion points or underutilized resources; otherwise, adjustments or retraining may be needed to improve its performance. Based on the predictions, the module communicates with the network controller to proactively adjust routing policies, allocate bandwidth, or reroute traffic flows, ensuring optimal performance and preventing traffic bottlenecks. Additionally, it periodically updates its model with new data to refine future predictions and adapt to changing traffic dynamics.

IV. Prediction Module and Methodology

In this paragraph, we proceed with the description of the steps that we followed to train and test our prediction model.

A. Data Collection and Preprocessing

We started by the generation of a full mesh SDN network composed of 10 hosts and 10 OpenFlow switches using the Mininet emulator [12] and the SDN controller ONOS [13]. We then configured the flows to be generated by defining their packet size, their sending frequency and their duration. Having 10 hosts, 5 source-receiver pairs were created randomly, and a process/thread was considered between each pair to generate the flows. Each thread chooses a random number of flows (between 100 and 150) that will be exchanged between each pair. Once this number is established, it is randomly divided into the three flow types that can be low, medium and high. The flow type establishes the characteristics of the flow itself, namely duration, packet size and transmission speed. To avoid that all the flows of the same pair start at the same time, a delay was introduced with a probability from 1 to 8 that is assigned to the start of each subsequent flow. The open source Distributed Internet Traffic Generator (D-ITG) software was used to generate network traffic in a customizable and realistic way [14]. Regarding the data collection, we relied on a separate thread, which is based on 3 main functions that coordinate to provide switches ports statistics. The first function returns information about the traffic passing through all the ports of each switch in the network. The second function allows to select from the data provided by the previous function those that are of interest for the prediction module (i.e., the number of bytes received for each port). The third function takes as input, the time interval set for the analysis, the current number of bytes received and the number of bytes received previously. The execution of the 3 functions was repeated every 10 seconds where all the switches were queried to provide the traffic data relating to each port. The relevant information was then selected and the throughput was calculated by making the difference between the current number of bytes received and the number of bytes received in the previously established interval. The throughput values of each port of each switch were collected at each iteration and saved in specific files. The data collection thread ended its execution when all the flows were received for each pair of hosts. At that point, we proceeded with the analysis of data to extract information on the traffic flows, the protocols used and the characteristics of the packets. In fact, the data collected includes several information about the network topology, consisting of nodes (hosts, switches, routers), links and their characteristics (bandwidth, latency).

Once the data is converted to .csv format, it is possible to use specific tools such as line charts, histograms and scatter plots to visualize the evolution of the parameters over time, but also descriptive statistics tools to calculate means, standard deviations, minimums and maximums for the flow parameters, to analyze the correlations between different parameters to identify causal relationships, or for clustering, that is, grouping flows based on similar characteristics, or even identifying anomalous behaviors in network traffic. In our case, the collected data was examined to perform a clustering operation using an SVM algorithm.

B. Prediction Model

The Support Vector Machines (SVMs) [15] are a set of supervised learning methods used for classification and regression. SVMs work by mapping a dataset into a multi-

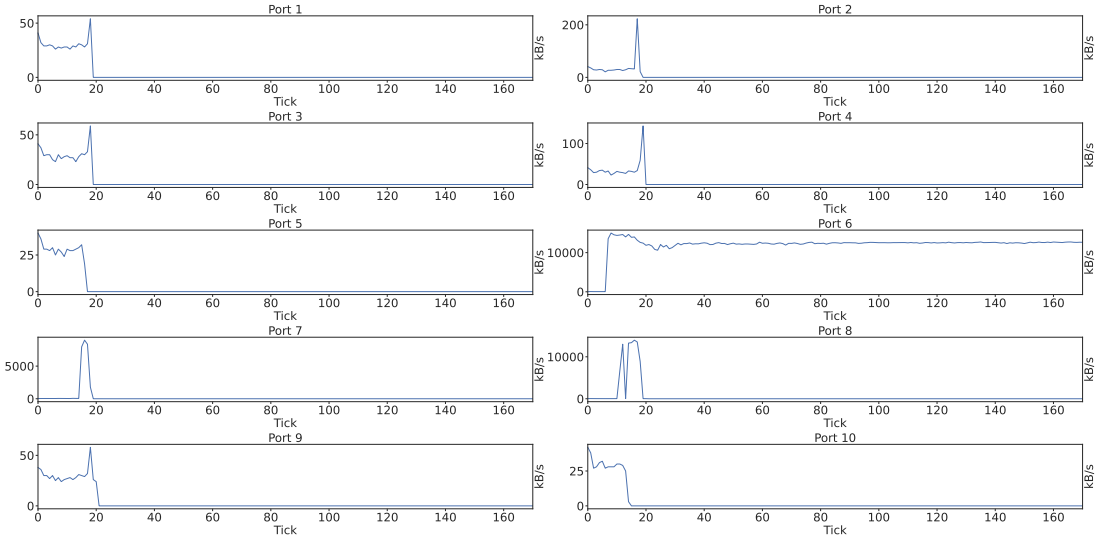

Fig. 3. Throughput of all the ten ports of a single switch during five hours of simulation with a sample every ten seconds.

dimensional space and finding the hyperplane (i.e., a ”line” that spans multiple dimensions) that best divides the data into classes. The optimal hyperplane is the one that maximizes the margin between classes. The margin is defined as the distance between the hyperplane and the data points in both classes that are closest to it. These closest data points are known as support vectors and are a subset of the training data. Once the hyperplane is identified, new data is classified by mapping it into the same space and determining which side of the hyperplane it lies on, essentially assigning the new data to one of the two classes, resulting in a linear classifier. For complex, non-linear data, SVM uses kernels, which are mathematical functions that transform the data into a higher-dimensional space where it becomes linearly separable. The choice of the kernel is one of the most important decisions in the application of SVMs. The kernel determines the space in which the data is mapped and, consequently, the complexity of the model and its ability to capture nonlinear relationships between the data, the choice of the optimal kernel strongly depends on the characteristics of the data and the specific problem. Common kernels include the linear kernel (for linearly separable data), the polynomial kernel, and the radial basis function (RBF) kernel, which is often used for more intricate, non-linear patterns [16]. The training process of an SVM involves solving a quadratic optimization problem with constraints. The goal is to find the vector of weights and biases that maximize the margin by correctly classifying the training data.

C. Training and performance Evaluation

In this paragraph, we proceed with the description of the training and testing processes of our case study. The data returned as output were written in a .csv file, that is, a table containing in the columns the throughput of each port of each switch. Each column is identified by the header ”switch n port n” and the rows are the values of the throughput t sampled every 10 seconds. Having channels with a capacity of $100bit/s$, we consider the network as congested if they are used for more than 70%. The values are inserted in an array X. A target array Y of boolean values is created that indicates 1 (true) if the value is above the threshold and 0 (false) if the value is below the threshold. The pairs of values (x,y) are then divided into training set (70% of the pairs of values) and testing set (30% of the pairs of values).

The type of prediction model was then specified. In our work, we consider an SVC classification model with RBF kernel. The model was trained with the training data (x,y) and then to perform the prediction, the x values of the testing dataset were provided to the SVC model and the model returned the predicted y values. Finally, the accuracy of the trained model was analyzed by relating the predicted y values to the expected y values of the datatset via confusion matrix, classification report and accuracy score. Having trained and validated the model, the congestion prediction can then be performed for each port of each switch with a new dataset.

V. Preliminary Evaluation

To validate our model, we created a reference simulation using Python and exploiting the main features offered by the Mininet emulator. Mininet focuses its workflow by default by creating industrial grade virtual switches called OpenVSwitches (OVS). This analysis is made to be the most realistic as possible, without the usage of simulated operating systems and environments. As stated in Sec. IV-A, the model was analyzed and tested using 10 switches connected in a mesh topology, with 10 hosts connected to random chosen switch ports. In Fig. 4, an example topology used in the tests is shown. Fig. 3 shows the throughput of all the ports of a single switch during a simulation of five hours with a sample taken every ten seconds. Every switch was equipped with ten ports. The various switches were under different loads and the resultant information were used to predict their congestion. From our experimental tests, the system starts in a state in

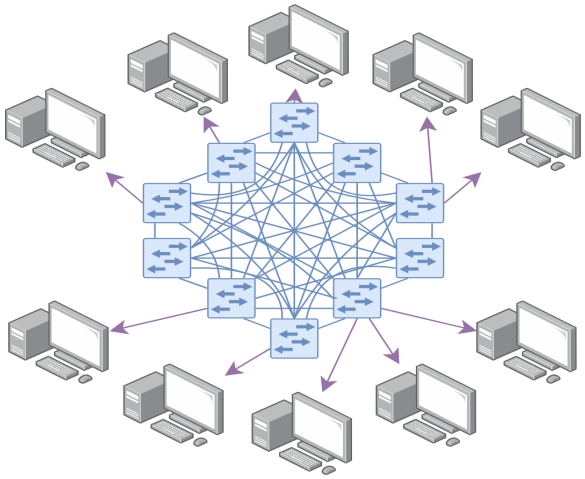

Fig. 4. Example of a topology analyzed by the simulation.

Accuracy	Precision	Recall	F1 Score
94.93%	100%	75.58%	86.09%
Confidence Interval 95%			
±1.32%	±0.1%	±1.13%	±0.94%

TABLE I
PERFORMANCE VALUES OF THE PREDICTOR MODEL.

which most of the ports send traffic to the other switches and nodes. Around tick 20, the system suffers from a spike of reconfiguration, in which all the ports start to send data. After this tick, the system is in its full-capacity, in which it selects a single port (e.g., Port 6 in Fig. 3), using it as the main throughput port. All the switches of the experiments exhibit the same behavior. As a preliminary evaluation of our approach, we present the confusion matrix in Table I, which is a tool used in machine learning to evaluate the performance of a classification model. By analyzing its values and calculating the appropriate metrics, the behavior of the model is evaluated. The confusion matrix compares the predictions made by the model with the actual results, allowing to calculate several metrics such as the precision, recall, F1-score, and accuracy. The resulting confusion matrix shows that the trained model always classifies correctly when the network is not congested (i.e., the precision is equal to 100%), while in the case of a congested network the model shows a recall value equal to 75.58%. F1 score ie equal to 86.09% indicating a good balance between precision and recall. Finally, the overall accuracy of the model is equal to 94.93% calculated over ten tests of five hours of run, which shows that the model correctly classifies most of the samples and behaves well.

VI. CONCLUSIONS

In this paper, we presented an SVM model for traffic prediction within the assurance operations of an IBN framework. The proposed model addresses the need for proactive network management by accurately forecasting traffic patterns, enabling early identification and mitigation of potential congestion. By leveraging SVM's capability to handle non-linear relationships, the model demonstrated high predictive accuracy thus enhancing the responsiveness and adaptability of the IBN framework in maintaining QoS and compliance with SLAs. Future work will explore extending this approach to include additional machine learning techniques and larger datasets to further enhance prediction accuracy and operational scalability.

ACKNOWLEDGMENT

The authors would like to thank Dr. Santi Mastroeni for his dedicated technical support during the experimental activities of this study. This research was financially supported by Universitas Mercatorum as part of the AI4SAFE project funded under the FIN/RIC program (id: 23-FIN/RIC). Additional funding was partially provided by the European Union - Next Generation EU under the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.3, CUP J53C22003120001, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART"). The Work was also carried out within the Department of Excellence in AI and Robotics 2023-2027 of Scuola Superiore Sant'Anna.

REFERENCES

[1] A. Leivadeas and M. Falkner, "A survey on intent-based networking," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, 2023.
[2] C. Casetti *et al.*, "Network slices for vertical industries," in *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, April 2018, pp. 254–259.
[3] A. Clemm *et al.*, "Intent-based networking - concepts and overview," in *[online] Available: https://bit.ly/2ImukBF*, January 2020.
[4] M. Gharbaoui *et al.*, "Intent-based networking: Current advances, open challenges, and future directions," in *2023 23rd International Conference on Transparent Optical Networks (ICTON)*, 2023, pp. 1–5.
[5] S. Barzegar *et al.*, "Intent-based networking for zero-touch optical networking," in *23rd International Conference on Transparent Optical Networks (ICTON)*, 2023, pp. 1–4.
[6] K. Abbas *et al.*, "AI-driven data analytics and intent-based networking for orchestration and control of B5G consumer electronics services," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, 2024.
[7] M. Sarwar *et al.*, "Ibn@cloud: An intent-based cloud and overlay network orchestration system," *Journal of Communications and Networks*, vol. 26, no. 1, pp. 131–146, 2024.
[8] J. Cunha *et al.*, "Enhancing network slicing security: Machine learning, software-defined networking, and network functions virtualization-driven strategies," *Future Internet*, vol. 16, no. 7, 2024.
[9] E. Coronado *et al.*, "Zero touch management: A survey of network automation solutions for 5g and 6g networks," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2535–2578, 2022.
[10] B. Martini *et al.*, "Intent-based network slicing for sdn vertical services with assurance: Context, design and preliminary experiments," *Future Generation Computer Systems*, vol. 142, 2023.
[11] ——, "Intent-based zero-touch service chaining layer for software-defined edge cloud networks," *Computer Networks*, vol. 212, 2022.
[12] Mininet official site. http://mininet.org/.
[13] "Onos project official site," https://onosproject.org/.
[14] A. Botta *et al.*, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, 2012.
[15] A. Mammone *et al.*, "Support vector machines," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 3, 2009.
[16] J. Cervantes *et al.*, "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, vol. 408, 2020.