IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 3 August 2022, accepted 17 August 2022, date of publication 26 August 2022, date of current version 6 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3201823

RESEARCH ARTICLE

Modeling and Control of Priority Queueing in Software Defined Networks via Machine Learning

ENRICO RETICCIOLI[1], GIOVANNI DOMENICO DI GIROLAMO[1], FRANCESCO SMARRA[1], ANGELO TORZI[2], FABIO GRAZIOSI[1], (Member, IEEE), AND ALESSANDRO D'INNOCENZO[1], (Member, IEEE)

[1]Department of Information Engineering, Computer Science and Mathematics, Università degli Studi dell'Aquila, 67100 L'Aquila, Italy
[2]Sonicatel s.r.l., 65127 Pescara, Italy

Corresponding author: Enrico Reticcioli (enrico.reticcioli@univaq.it)

This work was supported by the Italian Government through the Project "INnovating City Planning through Information and Communication Technologies (INCIPICT)" under Grant Cipe resolution 135 (December 2012).

ABSTRACT Software Defined Networking (SDN) is a new architectural paradigm that enables programmable control of a network to make it more flexible and easier to manage. SDN architectures decouple control and forwarding functionalities, and enable switches and routers to be remotely configurable/programmable in run-time by a controller. Modeling and optimization of such modern heterogeneous network infrastructures are key factors to achieve better performance, e.g. in terms of traffic flow improvement while reducing bandwidth allocation. Identifying an accurate model of a network device in SDNs (e.g., a switch or a router) is crucial in order to apply advanced techniques such as Model Predictive Control (MPC). However, such a problem is very challenging due to non-linearities and unavailability of internal variables measurements in real devices. To this aim, a promising direction is given by an appropriate integration of System Identification and Machine Learning techniques to obtain predictive models using historical data collected from the network thanks to the SDN paradigm. In this paper we apply a novel data-driven methodology to learn accurate models of the dynamical input-output behavior of a network's switch device by appropriately combining AutoRegressive eXogenous (ARX) model identification with Regression Trees (RTs) and Random Forests (RFs). The advantage of such model is that it can be directly used to apply MPC (which just requires Quadratic Programming to be solved) to optimally control the queues' bandwidth of the switch ports within the SDN paradigm. We validate our approach on an experimental emulation setup using the Mininet network emulator environment and a real dataset obtained from measurements of an Italian Internet Service Provider (Sonicatel S.r.l.). To this aim, we first develop a model of a real network switch, then implement MPC using the RYU controller, and finally demonstrate the benefits of the proposed dynamic queueing control methodology in terms of packet losses reduction and bandwidth saving, i.e. in terms of improvement of the Quality of Service.

INDEX TERMS Software defined networking, machine learning, system identification, network optimization.

I. INTRODUCTION

Communication networks involve the interconnection of a large number of devices, protocols and applications as well as specific Quality of Service (QoS) and Quality of Experience (QoE) requirements. The heterogeneity and complexity of such infrastructures pose a number of challenging problems in terms of modeling, managing and optimizing network resources (see e.g., [1], [2], [3], and references therein). In this scenario computing technologies, such as graphic and tensor processing units, represent a good opportunity to implement advanced control and machine learning techniques, such as Model Predictive Control (MPC), Decision Trees, Neural Networks, etc., in communication networks [4], [5], [6], [7].

The associate editor coordinating the review of this manuscript and approving it for publication was Seifedine Kadry.

Indeed, Machine Learning (ML) approaches applied to communication networks have been widely investigated over the past years. As an example, in [8] a Knowledge Plane (KP) approach has been proposed to enable automation, recommendation and intelligence by applying ML and cognitive techniques. However, the KP approach has been neither prototyped nor deployed because each node of traditional network systems, such as routers or switches, can only view and act over a small portion of the system. This implies that each node can learn only from a (small) part of the complete system. Thus, as shown in [9], it is very complex to design control algorithms beyond the local domain.

More in general, the infrastructure complexity, also due to the fact that currently data planes and control logics are packed together, brings to a very hard management of the networks. This makes actions like network configuration, with predefined policies, or reconfiguration, to face network issues (e.g., faults, changes, etc.), arduous if not impossible.

In this context, during the last decade, the new paradigm of Software Defined Networking has been developed and started to spread [10], [11], [12], [13], [14]. Such paradigm is mainly based on the separation of the network control plane from the data plane, i.e. the underlying structure made by routers and switches to manage the traffic flow. In this way, network switches only have the simple task to forward traffic, while the control logic can be implemented by an independent controller. This features provide the capacity to monitor and collect, in real-time, data of the network state and configuration, as well as packets and flow-granularity information [15], [16].

This new architecture opens a wide range of opportunities in terms of network management and run-time control. One among them is to improve the characteristics of each network device through ML algorithms, thus giving the opportunity to optimize network efficiency and performance. In particular, it enables the possibility to learn dynamical network models starting from historical data collected from the network to be used for management and optimal control purposes. More in details, a SDN controller device can configure the forwarding state of each switch by using a standard protocol called OpenFlow (OF) [17]. Thanks to the OF *counter variables* (e.g., flow statistics, port statistics, queue statistics, etc.), the controller can retrieve information (feedback) from the network devices, and store/process them for modeling and optimization purposes [18].

On this new paradigm, a vast amount of research has already been done trying to exploit its advantages. For example, in [19] a predictive model for estimating QoS in order to detect the need for a re-routing strategy due to link saturation is introduced. However, such framework cannot be used to apply traffic optimization techniques. In [20] an initial effort is conducted to derive a general hybrid system framework to model the traffic flow in communication networks. In [21], using hybrid systems, the authors provided a first formulation and implementation of a mathematical and simulative environment to formally model the effect of router/link failures on the dynamics of TCP and UDP packet flows belonging to different end-user services (i.e., http, ftp, mailing and video streaming). However, even though hybrid systems are very effective in modeling a network of routers, using such framework to implement traffic optimization is out of question due to computational complexity issues. In [22] the authors focused on designing strategies for periodic updating of network models to maintain good performance despite the evolution of the real system.

It is clear from all these studies, and many others [23], [24], [25], [26], [27], that the use of historical network data to derive accurate dynamical models of communication networks that can be directly used to setup optimal control problems, such as segment routing and/or queue management, is a challenging open problem.

A. MAIN CONTRIBUTION

In this paper we tackle the above problem and provide a methodology, based on algorithms from Machine Learning in synergy with methods from Control theory, to construct an input-output model of a switch device compliant with the SDN OpenFlow paradigm: such model is constructed so that MPC can be directly applied to such device to optimize the Priority Queueing performance just via Quadratic Programming solvers. To the best of the authors' knowledge, the state of the art (discussed in detail in Section II) still lacks of methods in this respect. In particular, all previous approaches require to directly measure the internal queues status, while from SDN devices this is not possible as one can only measure the input and output traffic of each queue. Furthermore, in the SDN OpenFlow paradigm it is not possible to directly control the queues schedule, but it is only possible to control minimum and maximum bandwidth bounds associated to each queue. In summary, none of the approaches in the existing literature proposes an input-output model identification technique or a control approach that is directly applicable to the input-output signals available in the SDN OpenFlow paradigm, and there is no straightforward way to extend existing techniques to our setting. On this line, the main contributions of this paper can be summarized as follows:

1) We apply theoretical results that we recently developed in [28] to combine Control theory (ARX identification) and Machine Learning (Regression Trees and Random Forests) in order to derive an accurate input-output model of a switch device compliant with the SDN OpenFlow paradigm to predict the queues states inside the switch ports using input/output (traffic flows) historical data. Note that, as discussed in Section II, previous approaches need access to the queue buffer data to identify the model and, at least with commercial SDN hardware, this is not possible. We use our model to set up an MPC problem to optimally control queues scheduling via bandwidth allocation, and implement such controller in the RYU SW environment [29]: this guarantees that our technique is directly applicable

to a real SDN network without any need of SW integration;
2) We validate our techniques on an experimental setup consisting of a packet-level network emulation environment and real traffic data from network measurements provided by an Italian Internet Service Provider (Sonicatel S.r.l.). Having used such validation setup, we also guarantee that the control code developed in this paper can be directly applied on a real SDN network based on the OpenFlow paradigm;
3) We compare the predictive accuracy of our RT- and RF-based techniques with Neural Networks (NNs): we show that, while NNs provide a slightly better preditctive accuracy, they are practically useless for optimal control purposes via MPC as they require to solve non-linear optimization problems, while using our methodology (based on RTs and RFs) MPC can be efficiently solved via Quadratic Programming. We also show that our methodology quickly improves its performance as soon as new data are available, bringing to an improvement of both model accuracy and control performance.

The present work is based on the preliminary conference paper in [30], which has been improved and extended in the following aspects: (i) the effect of iterative model updates using on-the-fly new data in the prediction accuracy is demonstrated; (ii) the effect of predictive models of future incoming traffic is tested; (iii) the accuracy of the predictive models is validated over a real dataset obtained from network measurements of an Italian Internet Service Provider (Sonicatel S.r.l.).

B. PAPER ORGANIZATION

The paper is organized as follows. In Section II a survey on the related work is provided. In Section III the network emulation environment is illustrated. In Section IV the model identification technique together with the MPC problem formulation is addressed. Finally, in Section V the methodology validation is provided showing results in terms of prediction accuracy and control performance.

II. RELATED WORK

The problem of modeling, managing and optimizing resources in a heterogeneous communication network is a very challenging engineering problem because of its inherent complexity [1], [2], [31], [32], [33]. Indeed, one of the most difficult challenges to be addressed to apply optimization techniques to the management of such complex networks is to derive predictive models of the queues of the switch behaviour [34], [35], [36], [37], [38], [39]. Numerous studies have also been conducted to maximize the performance of the controller and OpenFlow switch of SDNs. In particular, in the analysis of the literature regarding traffic management in SDNs and priority queueing we can distinguish three main different approaches: heuristic, parametric, and model-based.

Heuristic approaches are those where the algorithms to identify queue traffic models and control the queue traffic are based on rule-of-thumbs and empirical approaches that do not take into account any particular model. In [40] a heuristic method is proposed to balance the packet load among queues in order to reduce packet losses, although it does not aim at providing an optimal solution. In [41] the authors provide a scheduling algorithm to handle the incoming data traffic by enqueuing packets into the corresponding queue based on priority: High Priority queue is dequeued first. In [42] the authors define multiple queues with different priority classes, which are used to prioritize VoIP packet based on delay, and the controller decides where to enqueue the packet based on delay and considering 5 different decision thresholds.

Parametric approaches are those where the control of queues is based on less heuristics and more methodological solutions. More precisely, one or more parameters that describe the QoS of an SDN are chosen, and an optimization is performed based on *static* models characterized by such parameters. In [43] and [44] the authors consider different approaches to model and control queuing delays with specific network parameters. In [45] QoE is taken into account in the context of VOIP, and the decision metric to select the best link to establish a new VoIP call is based on the MOS quality metric, which is a typical measure of the user's satisfaction level of the quality of a call. All these approaches, despite the fact that they are easy to understand and implement, may not be often suitable to describe and control traffic flows in large and complex networks, as they are not based on network dynamical models.

Model-based approaches are those where a mathematical model of packet flows within a queue is considered. In classical literature of queuing theory, in particular applied to SDNs, most of the approaches are based on classical modeling structures (see e.g., [46]), and many techniques are exploited to estimate the parameters and the state of a queue [47]. In [48] the authors emphasized that switch performance depend on multiple factors, such as flow-table size, packet arrival rate, etc., and they took these key factors into account to design their M/Geo/1 system where the arriving packets follow a Poisson distribution and the service times follow a Geometric distribution. In [49], beside describing a comprehensive review of the literature (mostly M/M/k and M/G/k, i.e. specific queue models), the authors derived a new model for queueing networks based on Quasi Bird Death (QBD) processes. Another approach based on dynamical modeling for MPC is described in [50]. Here the authors derived a Discrete-time Markov Jump Linear System to model a queueing network with the aim of defining predictive control policies. In [51] the authors propose a new congestion control algorithm based on MPC, called MPAQM, where the queue length is predicted based on the extended TCP/AQM system model and on a state estimator. The main drawback of the approach proposed by the authors is that they linearize and discretize the model of a TCP/AQM interconnection system as in [52]: this is not always an effective

solution, since linearizing a model of a complex system does not always ensure adequate control performance, especially when the system is going to operate far from the linearization point. As an alternative nonlinear MPC can be applied, but the resulting optimization problem would be computationally intractable in general, and would not always guarantee a (global) solution.

Model-based methods are clearly the most related approaches with respect to the methodology proposed in this paper. For this reason, we want to provide a list of technical reasons that make our approach novel with respect to the state of the art:

- As already discussed in the previous section, the first and most relevant drawback of the existing model-based methods is related to the identification procedure of the queueing model: all of them need access to queue buffer data to identify/parameterize the model and, at least with commercial SDN hardware, this is not possible. In general, all the methodologies proposed in the related work do not allow to derive input-output models that enable direct exploitation of MPC while guaranteeing both good modeling accuracy and a tractable computational complexity (i.e., using Quadratic Programming (QP) solvers).
- The methods previously discussed are usually designed to provide best accuracy for a one step prediction. However, when dealing with MPC, a model to accurately predict the value of state variables over a finite future time horizon of arbitrary length N is needed. Classical approaches use the prediction computed at step $k+1$, and then iterate the model to predict the state at steps $k+2, \ldots, k+N$. However, this approach can suffer of several issues when N is large and the system is complex, such as numerical inaccuracies, error propagation, and additional uncertainties. In such situations, learning different models to predict the evolution of the state over different steps of the horizon (as we propose in this work) can increase MPC performance [28], [53]. Moreover, as we show later on in the paper, the additional computational complexity related to the number of models used for the prediction over the N steps of the horizon is negligible, thanks to the binary structure of the decision trees and to standard parallel computation techniques.

III. NETWORK EMULATION ENVIRONMENT

In this paper we use the Mininet environment [54] to emulate a SDN network to validate our methodology in terms of prediction accuracy and control performance.

This software runs a collection of virtual network elements (i.e., end-hosts, switches, routers, and links) on a single Linux kernel using lightweight virtualization. To generate traffic we use the Distributed Internet Traffic Generator (D-ITG) generator [55], [56], [57].

For the purposes of this work, we tested various network configurations: since similar results have been obtained on all

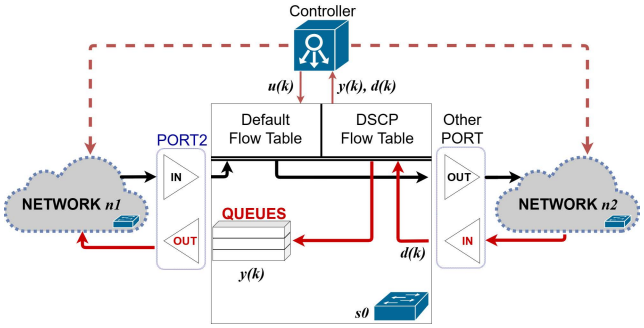


FIGURE 1. Mininet emulated network architecture. Packets generated by network *n2* are sent to network *n1* (red line) through switch *s0*. Each packet is sorted by the DSCP flow tables and sent to the appropriate queue on Port *2* according to the packet's priority level. Switch ports, represented by the white triangles, are for Input and Output.

configurations, we consider the generic case of the architecture depicted in Figure 1, which aims to represent a portion of a larger network where a bottleneck occurs. More precisely, we emulated a switch $s0$ with one input port and one output port, and a remote controller [29], [58], that dynamically manages the configuration of the queues of $s0$. The input of $s0$ is fed with more instances of D-ITG generating stochastic traffic, whose mean value follows the pattern of a real dataset (where packets are differentiated by their Type of Service (ToS) priority index) extracted from two days of logs of a large service provider network router. The original real dataset contains traffic data of a real network incoming from a source geographic area and terminating in a destination geographic area, and is divided for each value of Differentiated Services Code Point (DSCP) with a sampling time of 5 minutes [59], [60]. As depicted in Figure 1 D-ITG generates traffic from both networks, but only the packets generated from $n2$ to $n1$, represented by the red arrow, are sorted into the queues in $Port2$ of $s0$ according to the rules set by the flow tables. The switch ports are for both Input and Output, and are represented by the white triangles.

DSCP is the modern definition of the ToS field in which the first 6 bits are the Differentiated Services field that are in common with ToS field, while the last 2 bits regard explicit congestion notification. The ToS field can specify the priority of a datagram and the request for a low delay addressing a high throughput or a high reliability service. Following the implementation of many national service provider networks (see e.g., [61]), the 8 different values of the DSCP has been partitioned in three classes, and thus their relative packets are routed in three different queues with a different priority: the *Default* class (DSCPs 0, 1, 3) with a queue bandwidth of 20% of the port bandwidth, the *Premium* class (DSCPs 2, 4, 6, 7) with the 80% of the port bandwidth, and the *Gold* class (DSCP 5) with the 100% of the port bandwidth in a preemptive mode. This static implementation described in technical report [61] depicted in Figure 2, is the one used by Telecom Italia in real networks and is thus the one we consider to compare with.

We used the D-ITG Sender and Receiver modules to establish a connection between networks *n1* and *n2*. In particular, *16* D-ITG modules have been initialized: 8 for each network, and one for each DSCP index within each network. These modules handle the sampling time interval (5 minutes), the inter-departure time stochastic distribution associated with the packet rate, the packet size stochastic distribution, the IP and port destinations, and the DSCP index.

On the controller side we used RYU, which provides software components with well defined Application Programming Interfaces (APIs) that give the possibility to easily create new network management and control applications, and supports various protocols for managing network devices, such as OpenFlow, Netconf, OF-config, etc. About OpenFlow, RYU supports fully 1.0, 1.2, 1.3, 1.4, 1.5 and Nicira Extensions: for the considered test-bed, the 1.3 version has been chosen. In particular, we used the APIs in [62] and [63] for queue control and counter recovery from the switches. The feedback information collected for the purposes of this work are the descriptions of switches, ports and queues, the number of packets received and transmitted on each port of a switch, the packets passing through the flow tables, the packet rate values of each queue, and the packets transmitted by each single queue. In summary, the variables associated to the traffic and control signals in the closed-loop architecture, at each time step k, are the following:

- $d(k) \in \mathbb{R}^{10}$ is a measurable disturbance vector, i.e. representing variables that cannot be controlled. The first 8 components $d_1(k), \ldots, d_8(k)$ consist of the number of packets incoming in the switch $s0$ differentiated with respect to 8 different values of DSCP numbers. $d_9(k)$ and $d_{10}(k)$ are proxy variables, i.e. the hours and minutes of the day, which are very useful to the predictive model since traffic dynamics are tightly correlated with them, e.g. they are substantially different between night and day;
- $y(k) \in \mathbb{R}^3$ is the measured output vector, i.e. the variables to regulate. They consist of the number of packets outgoing from switch $s0$ differentiated with respect to the corresponding service class: $y_1(k)$ is the Default Queue output, $y_2(k)$ is the Premium Queue output and $y_3(k)$ is the Gold Queue output;
- $u(k) \in \mathbb{R}^3$ is the control input vector. Each component corresponds to the queue configuration of each service class: $u_1(k)$ is the Default Queue configuration, i.e. the maximum admitted bandwidth; $u_2(k)$ is the Premium Queue configuration, i.e. the maximum admitted bandwidth; $u_3(k)$ is the Gold Queue configuration, i.e. the minimum admitted bandwidth;

In this work the static control of the queues used in the Italian service provider network of *Telecom Italia* [61] has been first applied in the chosen emulative scenario, which is depicted in Figure 2. To this aim, 3 queues in $s0$ have been defined and configured as follows: packets with the DSCP values 0, 1 and 3 (Default queue) are routed via queue 0, with maximum rate $u_1(k) = 20MB/s, \forall k$; packets with values 2, 4, 6 and 7 (Premium queue) are routed on queue 1, with maximum rate $u_2(k) = 80MB/s, \forall k$; packets with value 5 (Gold queue) are routed on queue 2, with minimum rate $u_3(k) = 100MB/s, \forall k$. To obtain this prioritization it has also been necessary to set the flow tables of $s0$ to discriminate incoming packets based on the DSCP value and the destination IP address, and re-route them to the desired queue. Moreover, to obtain a bottleneck situation in $s0$, the bandwidth of the output port of switch $s0$ has been set to 100 *MB/s*, also to respect the limitation of Mininet in terms of CPU usage. Using this configuration, queue 2 uses the maximum capacity of the port to forward packets with preemptive priority, while the other two queues use the remaining bandwidth from 0 *MB/s* to the specified maximum bandwidth based on needs.

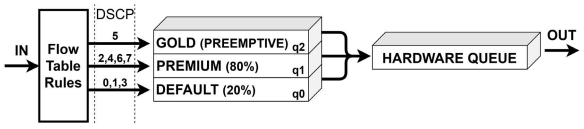


FIGURE 2. Telecom Italia [61] static queues rate in terms of percentage of the total amount of MB/s for a SDN switch port and packets identified by their DSCP numbers and routed relatively to their priority through each queue.

As will be shown in Section V, using static priority control the queues will not be able to send all the packets incoming from network $n1$, and a dramatic amount of packets will be lost. This motivates the application of optimization techniques, which are enabled by the predictive models derived using the methodology described in the following section.

In order to not to degrade the performance of the SDN network with the computation of the prediction model and of the control inputs at each instant k, the use two different computers has been chosen, as shown in Figure 3. In PC1 the python scripts useful for the establishment of the SDN network, the generation of a traffic pattern based on real data and the RYU controller are implemented. The latter takes care of saving in a local folder all the data useful for the creation of the prediction model by adding the measurements made at instant k inside specific files. Once the dataset is sufficiently populated, PC2 with Matlab software is used to compute a model with adequate prediction accuracy. This model is then used in real time to compute, and immediately apply, the control input $u(k)$ at each measurement instant k starting from the knowledge of the incoming traffic ($d(k)$) and of the outgoing packets from the queues ($y(k)$). The exchange of this information between Matlab and RYU is carried out through files saved on a shared folder on the internet (Dropbox).

IV. SWITCHED AFFINE MODELING VIA RTs AND RFs

In this section, we introduce a methodology to apply the results proposed in [28] and [64] to identify, starting from a set of collected historical data $\mathcal{D} = \{y(k), u(k), d(k)\}_{k=0}^{\ell}$, a switching ARX model of the input-output behavior of the traffic flow in a switch of an SDN network. Such model has

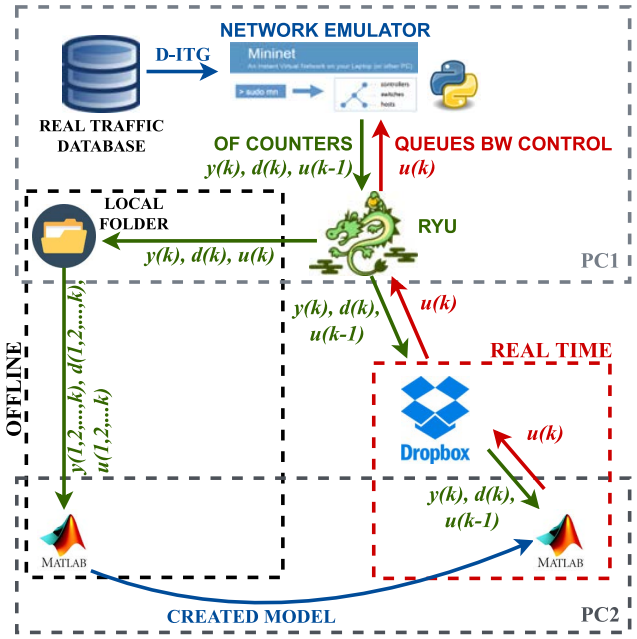


FIGURE 3. Interactions between PC1, where SDN Network resides, and PC2, that is responsable of offline calculation of the forecast model. The information exchange between Matlab and RYU is carried out through files saved on a shared folder on the internet (Dropbox).

the following form:

$$\begin{aligned} x(k+j+1) = {} & A'_{\sigma_j(x(k),d(k))}x(k) \\ & + \sum_{\alpha=0}^{j} B'_{\sigma_j(x(k),d(k)),\alpha}u(k+\alpha) \\ & + f'_{\sigma_j(x(k),d(k))}, \end{aligned} \tag{1}$$

$j = 0, \ldots, N-1$, where $x(k) \doteq [y^\top(k) \cdots y^\top(k-\delta_y)\ u^\top(k-1) \cdots u^\top(k-\delta_u)]^\top \in \mathbb{R}^{n_x}$ is the extended state that characterizes the switching ARX model, $x_\iota(k) \doteq [y_\iota(k) \cdots y_\iota(k-\delta_y)\ u^\top(k-1) \cdots u^\top(k-\delta_u)]^\top \in \mathbb{R}^{\delta_y+1+3\delta_u}$, $\iota = 1, 2, 3$, and $\sigma_j{:}\mathbb{R}^{n_x+10} \to \mathcal{M} \subset \mathbb{N}$ is a switching signal that associates an operating mode in a finite set $\mathcal{M}$, to be computed via RTs and RFs, to each pair $(x(k), d(k))$ and each prediction step j of the horizon. N is the chosen predictive horizon, and represents the number of steps of the prediction in terms of sampling time instants. Its value in terms of time depends on the choice of the sampling time, e.g. in Section V we have a dataset of measurements sampled at 5 minutes, thus with a choice of $N = 5$ we are considering a prediction over an horizon of 25 minutes. Once identified, model (1) can be used to setup the following optimization problem, which can be solved using standard Quadratic Programming (QP):

Problem 1:

$$\underset{u_0,\ldots,u_{N-1}}{\text{minimize}} \sum_{j=0}^{N-1} \left(\left(x_{j+1} - x_{\text{ref}}\right)^\top Q \left(x_{j+1} - x_{\text{ref}}\right) + u_j^\top R u_j\right)$$

subject to

$$\begin{aligned} x_{j+1} = {} & A'_{\sigma_j(x_0,d_0)}x_j \\ & + \sum_{\alpha=0}^{j} B'_{\sigma_j(x_0,d_0),\alpha}u_\alpha + f'_{\sigma_j(x_0,d_0)} \\ u_j \in {} & \mathcal{U} \\ x_0 = {} & x(k), \quad d_0 = d(k) \\ j = {} & 0, \ldots, N-1. \end{aligned}$$

As it is well known [65], Problem 1 is solved at each time step k using QP to compute the optimal sequence $u_0^*, \ldots, u_{N-1}^*$, but only the first input is applied to the system, i.e. $u(k) = u_0^*$. Note that, for any prediction step j, x_{j+1} only depends on the measurements $x_0 = x(k)$, $d_0 = d(k)$ at time k, since they are the only available measurements at time-step k. Matrices $Q \succeq 0$ and $R \succ 0$ are, as usual, chosen to trade-off the reference tracking and the control effort.

A. RTs AND RFs BACKGROUND

Let us consider a dataset $\{y(k), x_1(k), \ldots, x_\eta(k)\}_{k=0}^{\ell}$, with $y, x_1, \ldots, x_\eta \in \mathbb{R}$. Let us suppose to estimate, using Regression Trees, the prediction of the (response) variable $y(k)$ using the values of predictor variables $x_1(k), \ldots, x_\eta(k)$. The CART algorithm [66] creates an RT structure via optimal partition of the dataset. It solves a Least Square problem by optimally choosing recursively a variable to split and a corresponding splitting point. After several steps the algorithm converges to the optimal solution, and the dataset is partitioned in hyper-rectangular regions (the leaves of the tree) $R_1, R_2, \cdots, R_v$. In each partition $y(k)$ is estimated with a different constant $\hat{y}_i$, $i = 1, \ldots, v$, given by the average of the samples of $y(k)$ falling in R_i, i.e.

$$\hat{y}_i = \frac{\sum_{\{k|(x_1(k),\ldots,x_\eta(k))\in R_i\}} y(k)}{|R_i|} \tag{2}$$

Random Forests [67] are instead an averaging method that exploits a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The output prediction is given by averaging the predictions provided by all trees in the forest. It is possible to show that the error introduced by the forests quickly and almost surely converges to a limit as the number of trees in the forest becomes large. Such error also depends on the strength of the individual trees in the forest and the correlation between them. Thus, due to the Law of Large Numbers, RFs (differently from RTs) do not suffer much variance and overfitting problems. For more details the reader is referred to [66] and [67], while a more brief and concise description can be found in the Appendix of [53].

B. SWITCHING ARX (SARX) MODEL IDENTIFICATION VIA RTs

To derive a model as in (1), the main idea is to exploit the RT structure provided by CART to create a switching modeling framework. This can be done in two steps: (i) we split the dataset separating the control and non-control variables; (ii) we change the constant prediction provided by the

tree with a dynamical model in each leaf, hence creating a switching behavior.

To this aim, a new dataset $\mathcal{X} \doteq \{x(k), u(k), d(k)\}_{k=0}^{\ell}$ has to be defined starting from $\mathcal{D}$. In order to apply MPC, a model for each component of $y(k)$ that can predict system's dynamics over a horizon of length N is needed. To this aim, the idea is to create $3N$ predictive trees $\{\mathcal{T}_{\iota,j}\}$, $\iota = 1, 2, 3$, $j = 0, \ldots, N-1$, each one used to predict the 3 output components of the system over the N steps of the horizon. To create the tree structure the RTs algorithm (CART) partitions the dataset $\mathcal{X}$ into regions $\mathcal{X}_i$ such that $\biguplus \mathcal{X}_i = \mathcal{X}$, where $\biguplus$ denotes the disjoint union, and assigns to each region a constant value given by the average of the output values of the samples that ended up in that leaf. In run-time, once the trees are created, and given a real-time measurement $(x(k), u(k), d(k))$ belonging to leaf i, the CART algorithm provides as a prediction the averaged value associated to the leaf as in (2). However, since the prediction provided by the RT is a constant value, it cannot be used to setup an MPC problem, thus the learning procedure needs to be modified to identify a modeling framework as in (7). To this end, $\mathcal{X}$ is partitioned in two disjoint sets $\mathcal{X}_c = \{u(k)\}_{k=0}^{\ell}$, of data associated to the control variables, and $\mathcal{X}_{nc} = \{(x(k), d(k))\}_{k=0}^{\ell}$, of data associated to remaining variables. Then, the CART algorithm is applied only on $\mathcal{X}_{nc}$ (this is to avoid that the MPC problem turns out into a Mixed Integer Quadratic Program, see [28], [64] for details); thus, $3N$ RTs $\{\mathcal{T}_{\iota,j}\}$ have been created, each constructed to predict the variable $y_\iota(k+j+1)$, and consequently $x_\iota(k+j+1)$. In particular, to each leaf ι, i_j, corresponding to the partition $\mathcal{X}_{nc,\iota,i_j}$, of each tree $\mathcal{T}_{\iota,j}$, the following affine model is associated:

$$x_\iota(k+j+1) = A'_{\iota,i_j} x(k) + \sum_{\alpha=0}^{j} B'_{\iota,i_j,\alpha} u(k+\alpha) + f'_{\iota,i_j}, \quad (3)$$

where the coefficients of matrices A'_{ι,i_j}, $B'_{\iota,i_j,\alpha}$ and f'_{ι,i_j} in Eq. (4), as shown at the bottom of the next page, are obtained in each leaf ι, i_j by fitting the corresponding set of samples solving the Least Squares Problem 2.

Problem 2:

$$\begin{aligned} \underset{\xi_{\iota,i_j}}{\text{minimize}} \; & \|\Lambda_{\iota,i_j}\xi_{\iota,i_j} - \lambda_{\iota,i_j}\|_2^2 \\ \text{subject to } & f_{\min} \le f \le f_{\max} \\ & a_{\min} \le a_J \le a_{\max} \\ & b_{\min} \le b_{\iota,\alpha} \le b_{\max}, \end{aligned} \quad (5)$$

where ξ_{ι,i_j}, λ_{ι,i_j}, and Λ_{ι,i_j} contain respectively the parameters of matrices in (4), the predictions $x_\iota(k+j+1)$, and the current measurements of $x(k)$ and $u(k+\alpha)$. Linear inequalities (5) are used to constrain elements in ξ_{ι,i_j} to take into account physical system constraints and improve prediction accuracy. Model (3) can be easily compacted in the following form taking into account all the states $\iota = 1, 2, 3$:

$$x(k+j+1) = A'_{i_j} x(k) + \sum_{\alpha=0}^{j} B'_{i_j,\alpha} u(k+\alpha) + f'_{i_j}. \quad (6)$$

In particular, with the specific choice of $\delta_u = 0$ model (1) can be rewritten in the following state-space formulation

$$\begin{aligned} x(k+j+1) = & A_{\sigma_j(x(k),\mathbf{u}^-(k),d(k))} x(k+j) \\ & + B_{\sigma_j(x(k),\mathbf{u}^-(k),d(k))} u(k+j) \\ & + f_{\sigma_j(x(k),\mathbf{u}^-(k),d(k))}, \end{aligned} \quad (7)$$

where $\mathbf{u}^-(k) = [u^\top(k-1) \cdots u^\top(k-\delta)]^\top$ is the vector with the regressive terms of the input used to only grow the trees, and $\sigma_j : \mathbb{R}^{3(\delta_y+1)+3\delta+10} \to \mathcal{M} \subset \mathbb{N}$.

Thanks to this new formulation the following proposition shows that model (6) is equivalent to model (7) for any initial condition, any switching signal and any control sequence.

Proposition 1 [28]: Let A'_{i_j}, $B'_{i_j,\alpha}$ and f'_{i_j}, $\alpha = 0, \ldots, j$, $j = 0, \ldots, N-1$, be given. If A'_{i_j} is invertible for $j = 0, \ldots, N-1$, then there exists a model in the form

$$\begin{aligned} \bar{x}(k+j+1) = & A_{\sigma_j(\bar{x}(k),\mathbf{u}^-(k),d(k))} \bar{x}(k+j) \\ & + B_{\sigma_j(\bar{x}(k),\mathbf{u}^-(k),d(k))} u(k+j) \\ & + f_{\sigma_j(\bar{x}(k),\mathbf{u}^-(k),d(k))} \end{aligned}$$

such that for any initial condition $\bar{x}(k) = x(k) = x_k$, any switching sequence $i_0, \ldots, i_{N-1}$ and any control sequence $u(k), \ldots, u(k+N-1)$, then $\bar{x}(k+j+1) = x(k+j+1)$, $\forall j = 0, \ldots, N-1$.

As discussed in [28], from experimental findings it is possible to notice that the contribution in terms of model accuracy introduced by the choice of $\delta_u = 0$ is negligible, since previous control inputs are already considered in the tree structure choosing $\delta > 0$. Thus, in the rest of the paper we will consider $\delta_u = 0$ and $\delta > 0$.

C. SARX MODEL IDENTIFICATION VIA RFs

RF-based models can be constructed exploiting the RT-based formulation: in particular, let us consider $3N$ RFs $\mathcal{F}_{\iota,j}$, $\iota = 1, 2, 3$, $j = 0, \ldots, N-1$. For each tree $\mathcal{T}_\tau^{\mathcal{F}_{\iota,j}}$ of the forest $\mathcal{F}_{\iota,j}$, it is possible to estimate the coefficients a_*, b_* and f in (4) for each leaf ι, j, i_τ, i.e. $\tilde{\xi}_{\iota,j,i_\tau}$, solving Problem 2. With a small abuse of notation, let us indicate by $|\mathcal{F}_{\iota,j}|$ the number of trees in the forest ι, j. Then $\forall \iota, j$, the parameters to build matrices in (9) can be obtained by averaging parameters a_*, b_* and f, $\forall \tau = 1, \ldots, |\mathcal{F}_{\iota,j}|$, i.e.

$$\tilde{\xi}_{\iota,j} = \frac{\sum_{\tau=1}^{|\mathcal{F}_{\iota,j}|} \tilde{\xi}_{\iota,j,i_\tau}}{|\mathcal{F}_{\iota,j}|}, \quad (8)$$

over all the trees of forest $\mathcal{F}_{\iota,j}$. At this point, starting from (3), it can be easily construct the following model, as in (6) to be used in the MPC formulation by combining for $\iota = 1, 2, 3$ the matrices in (4) coming either from the RTs or from the RFs:

$$x(k+j+1) = A'_{i_j} x(k) + \sum_{\alpha=0}^{j} B'_{i_j,\alpha} u(k+\alpha) + f'_{i_j}. \quad (9)$$

D. MPC PROBLEM FORMULATION

We use model (9) to formalize Problem 1 according to the SDN priority queueing problem in Problem 3.

Problem 3:

$$\begin{aligned}
\min_{\mathbf{u}} \ & \sum_{j=0}^{N-1} \left[(x_{j+1} - x_{\text{ref},j})^\top Q (x_{j+1} - x_{\text{ref},j}) + u_j^\top R u_j \right] \\
\text{s.t. } & x_{j+1} = A_{\sigma_j(k)} x_j + B_{\sigma_j(k)} u_j + f_{\sigma_j(k)} \\
& \Delta u_\iota^{\min} \le u_{\iota,j} - u_{\iota,j-1} \le \Delta u_\iota^{\max} \\
& u_\iota^{\min} \le u_{\iota,j} \le u_\iota^{\max} \\
& u_{1,j} + u_{2,j} \le 100 \\
& x_0 = x(k), \quad \mathbf{u}_0^- = [u^\top(-1) \ \cdots \ u^\top(-\delta)]^\top, \\
& d_0 = d(k), \\
& j = 0, \ldots, N-1, \ \ \iota = 1, 2, 3,
\end{aligned}$$

where $\sigma_j(k) = \sigma_j(x(k), \mathbf{u}^-(k), d(k))$ (with a slight abuse of notation), $u_{\iota,j}$ is the ι^{th} component of the input u at time $k+j$; $\Delta u_\iota^{\min}$ and $\Delta u_\iota^{\max}$ are used to limit large variations on the inputs in 2 consecutive steps, in order to avoid that the queues drastically set to the minimum value and thus potentially increase packet losses during the next period; $u_\iota^{\min}$ and $u_\iota^{\max}$ define the bandwidth limits induced by the QoS requirements of the corresponding priority class. At each time step k the measurements of the variables in $\mathcal{X}_{nc}$ are collected, select the current matrices of (9) narrowing down the leaves of the trees, for $j = 0, \ldots, N-1$, solve Problem (3), and finally apply only the first input of the optimal sequence $\mathbf{u}^*$ found, i.e. $u(k) = u_0^*$.

As it is well known, Problem 3 represents a quadratic optimization problem, and its solution can be found with high efficiency with standard algorithms. This is a big advantage in terms of implementation on hardware platforms of our methodology, which is a future direction of this work. For this reason, we chose a quadratic function for Problem 3 as a good trade-off between control performance and computational complexity. However, the flexibility of the proposed methodology allows to change the objective function, for example with a linear, a nonlinear, or any other kind of function, clearly changing the trade-off between computational complexity and control performance. We refer the reader to [65] for further details on MPC complexity.

E. DISTURBANCE FORECAST

The knowledge at each time k of the future input traffic $(d(k+1), \ldots, d(k+N-1))$ can greatly improve the MPC performance. However, while the future values of the proxy variables (hours and minutes) are clearly well known, the knowledge of the future values of the first 8 component of the disturbance, i.e. number of packets incoming in the switches for each DSCP index are unknown at the current instant k. To address this problem, $8(N-1)$ RFs $\mathcal{F}_{\iota,j}^d$, $\iota = 1, \ldots, 8$, $j = 0, \ldots, N-1$ have been built in order to provide a prediction $\hat{d}_\iota(k+j)$ of the 8 disturbance components $d_\iota(k+j)$ over the future time horizon: as widely illustrated in [28], [64] the technique previously described can be easily modified by appropriately redefining the dataset as $\mathcal{X} = \{(x(k), u(k), d(k), \ldots, d(k+N-1))\}_{k=1}^{\ell}$ for the training phase, and considering a switching signal in (7) given by

$$A'_{\iota,i_j} = \begin{bmatrix}
a_1 & a_2 & \cdots & a_{\delta_y} & a_{\delta_y+1} & b_{\delta_y+2} & \cdots & b_{\delta_y+1+3(\delta_u-1)} & \cdots & b_{\delta_y+1+3\delta_u} \\
1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & 0 & \cdots & 0 & \cdots & 0 \\
0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & \cdots & 0
\end{bmatrix},$$

$$B'_{\iota,i_j,\alpha} = \begin{bmatrix} b_{1,\alpha} & b_{2,\alpha} & b_{3,\alpha} \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}, \ \alpha < j, \quad B'_{\iota,i_j,j} = \begin{bmatrix} b_{1,\alpha} & b_{2,\alpha} & b_{3,\alpha} \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}, \quad f'_{\iota,i_j} = \begin{bmatrix} f \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{4}$$

Algorithm 1 Data-Driven MPC With Regression Trees

Design time: Offline
Input: datasets $\mathcal{X}_{nc} = \{(x(k), d(k))\}_{k=0}^{\ell}$ and $\mathcal{X}_c = \{u(k)\}_{k=0}^{\ell}$
procedure Training LTI models in leaves
    Build $3N$ trees $\mathcal{T}_{\iota,j}$ using $\mathcal{X}_{nc}$, each to predict $y_\iota(k+j+1)$;
    for all $j = 0, \ldots, N-1$ **do**
        for all $\iota = 1, 2, 3$ **do**
            for all leaves $\iota, i_j \in \mathcal{T}_{\iota,j}$ **do**
                Compute ξ_{ι,i_j} by solving Problem 2;
            end for
            Build matrices $A'_{i_j}, B'_{i_j,\alpha}, f'_{i_j}$ in (4) using ξ_{ι,i_j};
        end for
    end for
end procedure

Run time: Online
Input: matrices $A'_{i_j}, B'_{i_j,\alpha}, f'_{i_j}, \forall\, i_j, \alpha$, constraint parameters $\Delta u_\iota^{\min}$, $\Delta u_\iota^{\max}$ $u_\iota^{\min}$, $\Delta u_\iota^{\max}$, and weight matrices Q,R
procedure Model Predictive Control via SARX-RT
    while $k \geq 0$ **do**
        for all $j = 0, \ldots, N-1$ **do**
            Using $\bar{x}(k), \mathbf{u}^-(k), d(k))$ narrow down each tree and determine ι, i_j;
            Determine $A_{\sigma_j}(k), B_{\sigma_j}(k), f_{\sigma_j}(k)$;
        end for
        Solve Problem 3 using QP to determine optimal inputs $u_k^*, \ldots, u_{k+j}^*$;
        Apply the first input $u(k) = u_k^*$;
    end while
end procedure

$\sigma_j(k) = \sigma_j(x(k), \mathbf{u}^-(k), d(k), \hat{d}(k+1), \ldots, \hat{d}(k+j)), \forall j = 0, \ldots, N-1$, i.e. also depending at time k on the future input traffic.

The whole procedure, consisting on both the offline model identification and the online control parts, has been summarized in Algorithm 1. In particular, the algorithm is given for the RTs, but it can be trivially generalized for the RFs.

V. SIMULATION RESULTS

In this section simulation results of the application of the proposed approach to SDN Priority Queueing identification and control is provided.

Standard RFs are used to derive models to forecast the disturbance components $d_1(k), \ldots, d_8(k)$, i.e. the switch input differentiated for each DSCP index, so as the predictive models for the output variable $y(k)$ following the approach described in Section IV. For both cases, the validation of the predictive accuracy is performed. In particular, the predictive models (based on RTs and RFs) are compared with Artificial Neural Networks, showing that RFs-based models represent the ideal solution both in terms of prediction accuracy and computational complexity. Then, the effect of iterative dataset updates in the prediction accuracy, both with and without forecast of the disturbances, is illustrated. Finally, the proposed predictive models are used to setup an MPC problem (Problem 3), and the control performance in terms of packet losses reduction and bandwidth saving, both with and without forcast of the disturbances, are investigated.

Results will show that, as expected, using accurate predictive models in the proposed MPC framework provides an important reduction of packet losses and an increase of bandwidth saving with respect to the static bandwidth allocation policy used in the Service Provider Networks as described in Section III: even thought this result is not surprising, it is very useful to show how better performance can be obtained in real networks only collecting historical data and applying a controller that can be directly implemented using the proposed identification procedure and Quadratic Programming solvers (which are well known to be very efficient).

In each of the aforementioned validations, 4 different predictive models have been exploited, using iteratively enriched datasets. As a training dataset we used a set composed by 63 days of data sampled every 5 minutes. In particular, to simulate a procedure that allows to update the models as soon as new data are available, we split the training dataset into 4 different subsets. We define by **OLD** the predictive model identified with a dataset consisting on the first 5124 collected samples (about 18 days), collected from network emulation with random values of the input $u(k)$. Then, we consider the availability of new measurements over time: by **1UP** (first update) we consider the predictive model identified with the **OLD** dataset enriched with 3456 new samples, i.e. a total of 8580 samples, about 29 days, obtained from network emulation when applying closed-loop MPC to define the input $u(k)$; by **2UP** (second update) we consider the predictive model identified with the **1UP** dataset enriched with 3168 new samples, i.e. a total of 11748 samples, about 40 days, obtained from network emulation when applying closed-loop MPC to define the input $u(k)$; and by **3UP** (third update) we consider the predictive model identified with the **2UP** dataset enriched with 6336 new samples, i.e. a total of 18084 samples, about 63 days, obtained from network emulation when applying closed-loop MPC to define the input $u(k)$.

As a testing dataset, we considered a different set composed by 6 days of data sampled every 5 minutes, i.e. 1684 samples, that we used to validate the learned models.

All simulations have been ran on a UDOO x86 Advanced with an Intel Braswell N3160 processor up to 2.24 GHz and 4 GB of RAM [68].

A. TRAFFIC PREDICTIVE MODEL VALIDATION ON EMULATION ENVIRONMENT

Having an accurate forcast of the variable $d(k)$ (i.e. the switch input differentiated for each DSCP index) can be helpful to improve the model identification performance, other than improving the tracking of the reference input $x_{\text{ref},j}$, $j = 1, 2, \ldots, N$ in Problem 3. In this section, the standard RFs

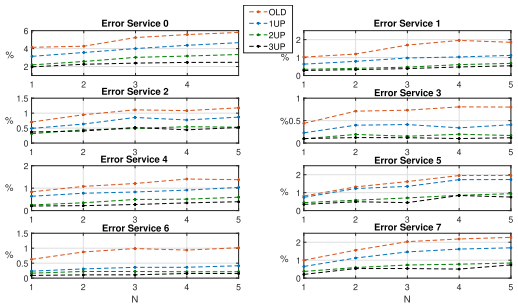


FIGURE 4. NRMSE of the disturbance predictive models over a time horizon of $N = 5$ (e.g. 25 minutes). Every model update improves the prediction accuracy for each DSCP service.

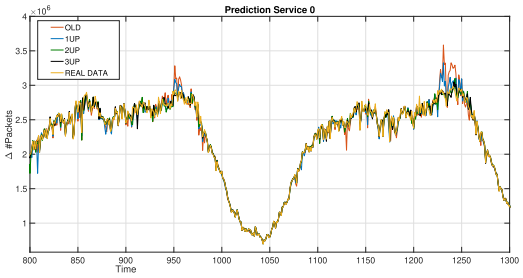


FIGURE 5. Comparison between Service 0 measured traffic (yellow line) and its traffic prediction for each model. The more a prediction overlaps the measured traffic, the more the prediction is accurate.

algorithm is applied with a regression of 15 on the variables d_i, and 30 trees for each forest, to obtain models to produce the disturbance forcasts over a predictive horizon of $N = 5$ (25 minutes). As intuitive, considering a large predictive horizon could generate a predictive model with low accuracy in terms of prediction, hence ruining the MPC performance. Furthermore, the larger N the higher the complexity of the closed-loop control, although the solution of the quadratic MPC would still be efficient. To test such hypothesis, we ran several simulations varying the value of N. Results confirmed our thoughts, thus we found N = 5, i.e. 25 minutes, to be a good trade-off between model accuracy and control complexity. Nevertheless, the choice of the value for N is strictly related to the quality and amount of data, and to the nature of the system under consideration, thus its best value depends on the system and on the control goal.

Figure 4 shows the Normalized Root Mean Square Error (NRMSE) of the models of the disturbance signals (one for each of the 8 DSCP indices) over a time horizon of $N = 5$: the prediction error is worse for Service 0 ($4 - 6\%$) since it includes the majority of the packets that transit through the switch. For other services the NRMSE is at most 2.2% (Service 7) over all the predictive horizons. The improvement of the model accuracy when using larger (updated) datasets is evident, until a *saturation* is reached and further data do not help to improve the model accuracy: the NRMSE significantly reduces and for Service 0 it is even halved. We chose the NRMSE metric to emphasize the deviation in percentage of our model predictions from the real measurements. Thus, the error normalization facilitates the comparison between datasets and models with different scales. Figure 5 shows, for Service 0 and in a time window of 500 samples (almost two days), the predictions of OLD, 1UP, 2UP and 3UP as well as the original data. In this case, the more packets prediction overlap the real data (yellow line) and the more the prediction is considered accurate. This result clearly highlights the better prediction accuracy provided with 2UP and 3UP with respect to OLD and 1UP.

1) TRAFFIC PREDICTIVE MODEL VALIDATION ON REAL NETWORK DATA

In addition to the validation of our predictive models of the incoming traffic over the Mininet environment, the accuracy has been also tested on data measured from a real network device (Ubiquiti EP-16) of an Italian internet provider (Soniticatel S.r.l.). Data collection has been performed using the software Cacti [69].

Since this device is part of a running commercial network, some constraints in data collection have forced to only measure the sum of all packets entering and leaving the device, and it has been possible to extract from such traffic only incoming VOIP packets, i.e. it has not been possible to extract packets differentiated for each DSCP. Moreover, it is not currently possible to apply any type of closed-loop control on the network device. For the above 2 reasons the control performance validation in the following sections is not based on this real traffic dataset.

For the validation procedure we split the dataset into a *training* set and a testing set. The training dataset consists of the first 53 days of data measurements (from 3^{rd} of March to 19^{th} of April 2020), that corresponds to 15264 samples, while the testing dataset consists of 3 days of data measurements (from 20^{th} to $22^{th}$of April 2020), that corresponds to 864 samples. We use the training set to learn RFs models and the testing set to validate the models. Figure 6 shows the prediction on three classes of packets: all packets received, all packets transmitted, VOIP packets received.

Although the errors grow quite rapidly, such results are promising towards the experimental direction. The first reason is that we can see how the errors are below the 10% at the first step, that is the most important prediction in terms of control, as the first input is the one that is applied during the control-loop. The second reason is that, as mentioned above, such results are obtained with a limited amount of data: we are currently extracting a larger dataset and expect much better performance when more data will be available, e.g. packets differentiated for each DSCP and this is material for future works.

B. QUEUES PREDICTIVE MODEL VALIDATION

In this section a comparison of the accuracy of RTs and RFs predictive models with Artificial Neural Networks is shown. A neural network is a collection of algorithms that identify underlying relations in a dataset: it consists of groups of connected neurons organized in layers, where the connections between neurons are modeled using weights. The signal

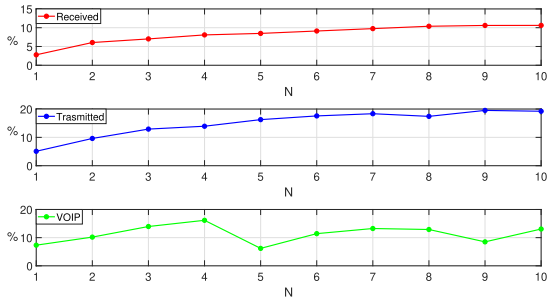


FIGURE 6. NRMSE of the packets predictive model over a time horizon of $N = 10$ (e.g. 50 minutes) with real dataset harvested from Italian internet provider (Sonicatel S.r.l.) network.

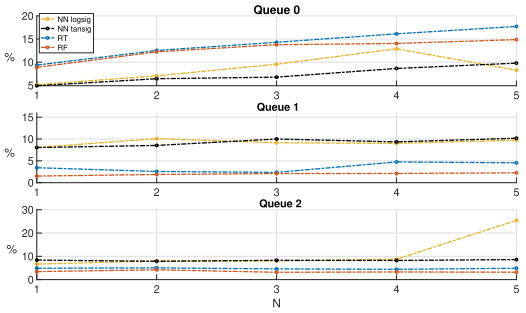


FIGURE 7. NRMSE over an horiozn of $N = 5$ and for each priority queue for RTs (blue), RFs (red), NN with sigmoids as activation function (yellow) and NN with hyperbolic tangent as activation function (black).

TABLE 1. Identification parameters.

Parameters	Value	Parameters	Values
N	5	f_{min}	-100
δ_y	1	f_{max}	100
δ_x	5	a_{min}	-100
δ_u	0	a_{max}	100
δ_d	5	b_{min}	0
δ	5	b_{max}	10000
$\lvert\mathcal{F}_{ij}\rvert$	30	Minleaf	13

produced with this linear composition is then fed into an activation function that is in general nonlinear. The reader is referred to [70] and references therein for more details. A wide number of tools to build Neural Networks have been developed during recent years, e.g. [71], [72], [73] just to mention a few: in this work it is exploited the Deep Learning Toolbox of Matlab to compare predictive models based on NNs with the methodology proposed in this paper, based on ARX combined RTs and RFs. It is considered here just OLD as the learning dataset and a predictive horizon $N = 5$.

To identify a RTs (resp. RFs)-based predictive model of the queues, a RT (resp. RF) is trained for each output and for each time horizon, with a total of 15 trees (resp. 15 forests each consisting of 30 trees). The main parameters used for the identification algorithm (see Section IV and Problem 2) are summarized in Table 1. In particular, as done with δ in Equation (7), parameters δ_x and δ_d are considered as regressive terms of the state and disturbance that will be only used to grow the trees and the forests, i.e. $\sigma_j(k) = \sigma_j(x(k), \ldots, x(k-\delta_x), \mathbf{u}^-(k), d(k), \ldots, d(k-\delta_d))$. The regressive terms ($\delta_d$, δ_y, δ_x, δ_u, δ) and the minimum number of samples for each tree of each forest (MinLeaf) have been chosen, with a trial and error approach, considering that very small regressive horizons and very large values for MinLeaf may lead to inaccurate prediction (as they do not provide sufficient information on the past) but very large regressive horizons and very small values for MinLeaf also lead to inaccurate prediction (as they interpolate very old data that might negatively affect the results and produce overfitting).

Regarding specific parameters used for running NN, and for the sake of a fair comparison, they have been tuned to obtain the best performance: in particular shallow networks of 2 layers are considered since deeper networks did not improve the accuracy and, instead, have the negative effect of increasing the sensitivity of the accuracy with respect to the initial conditions of the weights. The first layer with 20 neurons has 17 input, while the output layer has 3 neurons. Among the many algorithms for optimizing the weights of the neurons, the following will be considered: *Scaled conjugate gradient back-propagation* described in [74], which provided the best accuracy with respect to the dataset considered. Regarding the activation functions, both the classical sigmoid function (*LogSig*) and the Hyperbolic tangent sigmoid transfer function (*TanSig*) are used, while for the output layer a linear activation function has been used.

As a metric of the prediction accuracy in Figure 7 is shown a comparison between the Normalized Root Mean Square Errors (NRMSE) of the different identification approaches for each priority class and over a horizon up to $N = 5$. Regarding queue 0 (Default) NNs perform better than RTs and RFs, but in queues 1 (Premium) and 2 (Gold), characterized by higher priority, RFs provide the best performance. Queue 0 is characterized by a larger NRMSE with all identification techniques: this is due to the fact that, having the lowest priority, it suffers more packet losses and this can negatively affect the prediction accuracy. The validation emphasizes that RTs, even thought very simple and fast to compute, are often affected by overfitting and variance issues, i.e. small variations of the training data result in large variations of the tree structure and, consequently, of the predictions. Regarding NNs, they provide a less accurate model in 2 cases over 3. Indeed, by analyzing the dataset distribution, it is possible to notice a peculiar regular grid pattern that can be very well approximated by hyper-rectangles: since RTs and RFs base their prediction on hyper-rectangular dataset partitions, the better performance with respect to NNs is reasonable. For queue 0, even thought NNs perform better, it is important to remark that their predictive model is based on nonlinear functions: this makes the derived model impractical for real-time control as the corresponding MPC formulation

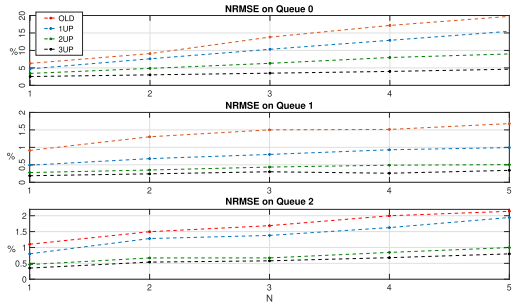


FIGURE 8. Comparison between NRMSE of the queues output predictive models over a time horizon of $N = 5$, without taking into account in the model computation the prediction of future disturbances.

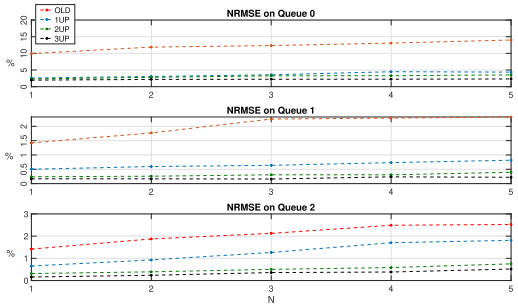


FIGURE 9. Comparison between NRMSE of the queues output predictive models over a time horizon of $N = 5$, including in the model computation the knowledge of the disturbance forecast over 4 steps (20 minutes) of the horiozn.

turns into a nonlinear optimization problem, for which there is no approach that can guarantee neither a global optimal solution nor a reasonable computation time. In addition to this, even obtaining a closed mathematical form of the predictive function of a Neural Network starting from neurons and weights is not always an easy task, because of the highly nonlinear interconnections between the different layers.

Remark 1: For all these reasons it will be only used, from now on, RF-based models which provide the best choice both from the accuracy and the computational complexity points of view.

Figure 8 and Figure 9 plot the NRMSEs respectively without and with prediction of the future disturbances showing the effect of iterative dataset updates. The assumption of future disturbance forecast, as expected, provides much better prediction accuracy. The positive effect of updated datasets is also clear, providing accuracy improvements up to 50%: as will be also discuss in the next section, the most relevant prediction accuracy improvement takes place moving from OLD to 1UP or from 1UP to 2UP, while the 3UP model does not improve much.

Remark 2: In the emulations shown in this work, data are generated without major modifications of the traffic daily pattern: for this reason enriching the dataset converges to a saturation of the model accuracy, as discussed above. Nevertheless, the capability of the proposed methodology to iteratively learn from new data is fundamental as, in real life, there are changes in traffic patterns, and model updates are necessary to maintain the model accuracy and the control performance.

TABLE 2. Constraints in Problem 3.

Parameters	Value	Parameters	Values
$\Delta u_1^{\min}$	1	$\Delta u_1^{\max}$	30
$\Delta u_2^{\min}$	20	$\Delta u_2^{\max}$	30
$\Delta u_3^{\min}$	20	$\Delta u_3^{\max}$	20
$u_1^{\min}$	10	$u_1^{\max}$	45
$u_2^{\min}$	55	$u_2^{\max}$	80
$u_3^{\min}$	80	$u_3^{\max}$	100

C. CONTROL PERFORMANCE

In this section a closed-loop control strategy is setup, where the (Mininet) network emulator and the (RYU) controller run in two different computers and synchronize/exchange data using a shared file. Namely, the SW controller module is ready to be directly used on a real SDN-based network, with just some minor modifications in the data exchange with the switch devices. The controller implements the optimal control strategy using the predictive models validated in the previous sections: at each time step, it solves Problem 3 and updates the bandwidth of the different queues. The cost matrices Q and R of Problem 3 respectively weight the output $y(k)$ of the system (i.e., the packet transmission rate for each queue) and the control input $u(k)$ (i.e., the bandwidth assigned to each queue). Since R is required to be positive definite, but it makes no sense to assign a penalty to the choice of $u(k)$, the choice has been $R = 10^{-5} \cdot \mathbb{I}$, where the identity matrix $\mathbb{I}$ multiplies a very small value. Matrix $Q = diag(1, 10^4, 10)$ has been assigned as a diagonal matrix, where the choice of the different diagonal components is related to the priority level of each queue. $N = 5$ has been chosen as the predictive horizon. The remaining constraints of Problem 3, reported in Table 2, have been selected to provide a queue bandwidth value range around the static values used by Telecom Italia [61] without damaging the QoS or the priority of each service. Such range is necessary in the optimization problem in order to allow for a trade-off in terms of performance. Clearly, the range can not be too large to avoid the risk to have a bad QoS or a bad priority of each service.

In what follows the control performance, both without and with forecast of the future disturbances, is validated. The values of $x_{\text{ref},j}$ in the optimization problem represent the reference values chosen for tracking system output: indeed, as the objective is to minimize packet losses, it is minimized the difference between the packets received by the hosts $d(k)$ and those transmitted by the queues $y(k)$ over the horizon N. In case there are no prediction of future disturbances, $x_{\text{ref},j} = x_{\text{ref}} = d(k)$ will be equal to the current disturbance measurement and constant over all the predictive horizon; if instead there is a prediction of future disturbances, $x_{\text{ref},j}$ will be equal to such forecast. In this section only models OLD, 1UP and

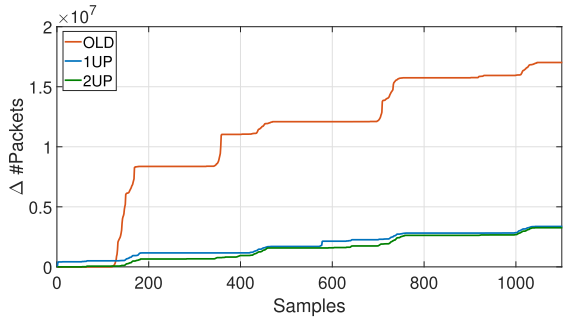


FIGURE 10. Comparison between Cumulative Packet Losses without knowledge of disturbance forecast for different models.

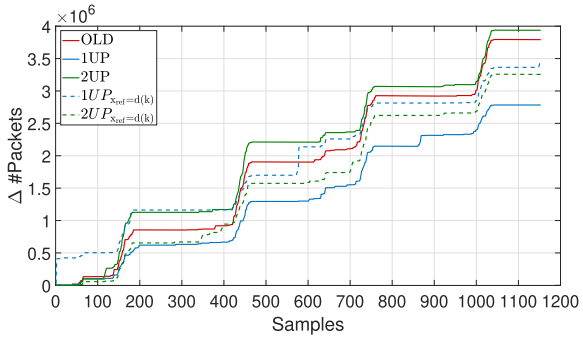


FIGURE 11. Comparison between Cumulative Packet Losses with (solid lines) and without (dashed lines) knowledge of disturbance forecast. If the disturbance forecast is not available, similar control performance can be obtained via one update of the model without the knowledge of disturbance.

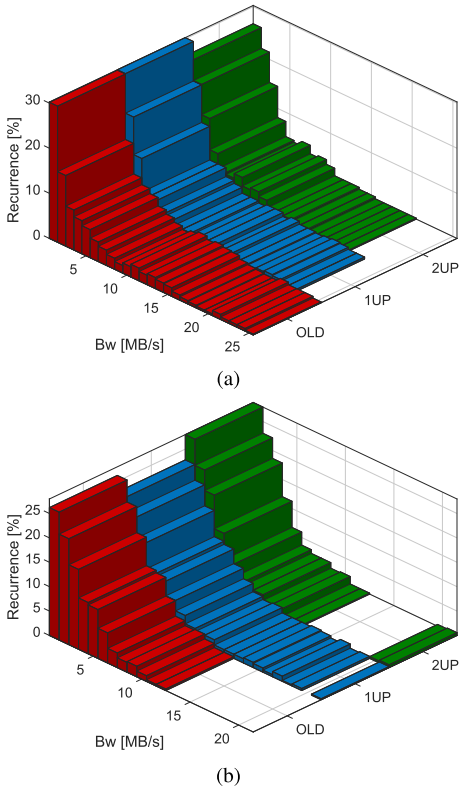


FIGURE 12. Bandwidth saving comparison without (a) and with (b) prediction of the future disturbance. Each slot represent the recurrence of the different bandwidth usage during the simulations.

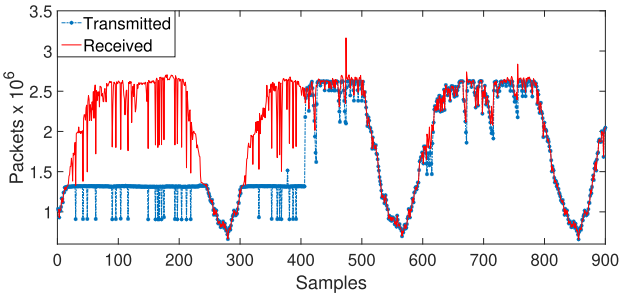


FIGURE 13. Red line shows incoming traffic, blue line shows sum of the packets sent from the queues. Their difference represents packet losses. The approach based on static queues is applied on the first 400^{th} samples (implemented as in [61]): many packet losses are generated due to queues saturation. Then MPC is activated: packet losses are drastically reduced.

2UP are compared, since model 3UP does not provide any substantial improvement.

Figures 10 and 11 plot the cumulative packet losses respectively without and with prediction of the future disturbances. The packet loss rate when the control is performed exploiting the OLD model and without disturbance forecast is around 123% larger than all other cases (and, of course, incomparably smaller than the static control case [61]). It is also clear from the plots that 1UP and 2UP without disturbance forecast and OLD, 1UP and 2UP with disturbance forecast provide very similar performance. Authors' interpretation is that OLD models without disturbance forecast have not enough information to provide good accuracy, but they can be easily improved either with a dataset update (which however requires 10 days for 1UP and 20 days for 2UP of additional data) or using a disturbance model. This means that the inclusion of disturbances prediction within the prediction model calculation provides optimal control performances even with relatively small training datasets, which supports the use of this type of control in real contexts.

Figure 12 illustrates the bandwidth savings showing the recurrence of the different bandwidth usage during the simulations, respectively without and with prediction of the future disturbances. Without disturbance forecast it is exploited up to $25MB/s$ using the OLD model, while it is exploited at most $22MB/s$ and $21MB/s$ respectively for models 1UP and 2UP. Using disturbance forecast, as expected, even less bandwidth is used.

In conclusion to this section, we show the gap between priority queueing control performance of MPC, obtained solving Problem 3 and based on the proposed RFs predictive model, with the static control policy adopted by service provider networks in [61]. Figure 13 highlights the dramatic improvement of MPC with respect to static control: the red line shows the incoming traffic, the blue line shows the sum of the packets sent from the queues, and their difference represents packet losses. Until the $400th$ sample static control

has been implemented as in [61], generating many packet losses due to queues saturation. From that sample to the end of the experimentation MPC is implemented using RF-based model, and the packet loss is drastically reduced: quantitatively, after 700 sampling periods the cumulative number of dropped packets with the static policy is about $5.5 \cdot 10^8$ versus $6.6 \cdot 10^6$ with MPC, with a decrease of $5.434 \cdot 10^8$ lost packets (-88%).

Even thought the improvement of the MPC strategy with respect to the static control is not surprising, much better performance can be obtained in real networks just collecting historical data and applying a controller that can be directly implemented using the accurate models of the proposed identification algorithms and Quadratic Programming standard solvers.

VI. CONCLUSION

In this paper a methodology to derive mathematical models for priority queueing in Software Defined Networks is presented and implemented, with the final aim of enabling the application of advanced optimization control techniques, such as MPC, in this context. The approach has been validated over the Mininet packet-level network emulator framework and using real data measured from the network of an Italian Internet Service Provider (Sonicatel s.r.l.). The control code has been implemented in the RYU SW environment: this guarantees that our technique is directly applicable to a real SDN network without any need of SW integration. The validation results have shown good performance in terms of prediction accuracy both for the incoming traffic flow and for the input/output behavior of a switch device in the proposed SDN-based setup. Such results have also provided promising insights on the potential impact of data-driven predictive models combined with MPC in terms of packet losses reduction and bandwidth savings in real networks. In future work, we plan to validate the control performance over a real network instead of using the Mininet emulator.

REFERENCES

[1] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synth. Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.

[2] O. Lemeshko, T. Lebedenko, and A. Al-Dulaimi, "Improvement of method of balanced queue management on routers interfaces of telecommunication networks," in *Proc. 3rd Int. Conf. Adv. Inf. Commun. Technol. (AICT)*, Jul. 2019, pp. 170–175.

[3] A. Mohan, A. Gopalan, and A. Kumar, "Reduced-state, optimal scheduling for decentralized medium access control of a class of wireless networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1017–1032, Jun. 2020.

[4] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Netw.*, vol. 32, no. 2, pp. 92–99, Mar. 2018.

[5] M. Usama, J. Qadir, A. Raza, H. Arif, K.-L. Alvin Yau, Y. Elkhatib, A. Hussain, and A. Al-Fuqaha, "Unsupervised machine learning for networking: Techniques, applications and research challenges," Sep. 2017, *arXiv:1709.06599*.

[6] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 393–430, 1st Quart., 2019.

[7] G. Xu, Y. Mu, and J. Liu, "Inclusion of artificial intelligence in communication networks and services," *ITU J., ICT Discoveries*, no. 1, Jan./Oct. 2017.

[8] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the internet," in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, 2003, pp. 3–10.

[9] A. Mestres *et al.*, "Knowledge-defined networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 3, pp. 2–10, Sep. 2017.

[10] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[11] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, Jul. 2013.

[12] M. Jarschel, T. Zinner, T. Hossfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, attributes, and use cases: A compass for SDN," *IEEE Commun. Mag.*, vol. 52, no. 6, pp. 210–217, Jun. 2014.

[13] T. Chen, M. Matinmikko, X. Chen, X. Zhou, and P. Ahokangas, "Software defined mobile networks: Concept, survey, and research directions," *IEEE Commun. Mag.*, vol. 53, no. 11, pp. 126–133, Nov. 2015.

[14] P. Ameigeiras, J. J. Ramos-Munoz, L. Schumacher, J. Prados-Garzon, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Link-level access cloud architecture design based on SDN for 5G networks," *IEEE Netw.*, vol. 29, no. 2, pp. 24–31, Mar. 2015.

[15] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, "Machine learning in software defined networks: Data collection and traffic classification," in *Proc. IEEE 24th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2016, pp. 1–5.

[16] Y. Zhai, H. Xu, H. Wang, Z. Meng, and H. Huang, "Joint routing and sketch configuration in software-defined networking," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2092–2105, Oct. 2020.

[17] N. McKeown, T. anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[18] The Open Networking Foundation. (2012). *OpenFlow Switch Specification, Version 1.3.0*. [Online]. Available: https://shorturl.at/ksH03

[19] M. Cello, M. Marchese, and M. Mongelli, "On the QoS estimation in an OpenFlow network: The packet loss case," *IEEE Commun. Lett.*, vol. 20, no. 3, pp. 554–557, Mar. 2016.

[20] J. Lee, S. Bohacek, J. Hespanha, and K. Obraczka, "Modeling communication networks with hybrid systems," *IEEE/ACM Trans. Netw.*, vol. 15, no. 3, pp. 630–643, Jun. 2007.

[21] M. D. Di Benedetto, A. Di Loreto, A. D'Innocenzo, and T. Ionta, "Modeling of traffic congestion and re-routing in a service provider network," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, Jun. 2014, pp. 557–562.

[22] P. Mulinka and P. Casas, "Stream-based machine learning for network security and anomaly detection," in *Proc. Workshop Big Data Anal. Mach. Learn. Data Commun. Netw.*, 2018, pp. 1–7.

[23] J. Kim and G. Hwang, "Prediction based efficient online bandwidth allocation method," *IEEE Commun. Lett.*, vol. 23, no. 12, pp. 2330–2334, Dec. 2019.

[24] W. Aljoby, X. Wang, T. Z. J. Fu, and R. T. B. Ma, "On SDN-enabled online and dynamic bandwidth allocation for stream analytics," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1688–1702, Aug. 2019.

[25] T. Lebedenko, O. Yeremenko, S. Harkusha, and A. S. Ali, "Dynamic model of queue management based on resource allocation in telecommunication networks," in *Proc. 14th Int. Conf. Adv. Trends Radioelecrtron., Telecommun. Comput. Eng. (TCSET)*, Feb. 2018, pp. 1035–1038.

[26] L. Le, J. Aikat, K. Jeffay, and F. D. Smith, "The effects of active queue management and explicit congestion notification on web performance," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1217–1230, Dec. 2007.

[27] S. Ghosh, R. Rajkumar, J. Hansen, and J. Lehoczky, "Scalable QoS-based resource allocation in hierarchical networked environment," in *Proc. 11th IEEE Real Time Embedded Technol. Appl. Symp.*, Mar. 2005, pp. 256–267.

[28] F. Smarra, G. D. Di Girolamo, V. De Iuliis, A. Jain, R. Mangharam, and A. D'Innocenzo, "Data-driven switching modeling for MPC using regression trees and random forests," *Nonlinear Anal., Hybrid Syst.*, vol. 36, May 2020, Art. no. 100882.

[29] (May 2019). *RYU Controller*. [Online]. Available: https://osrg.github.io/ryu/

[30] E. Reticcioli, G. D. Girolamo, F. Smarra, A. Carmenini, A. D'Innocenzo, and F. Graziosi, "Learning SDN traffic flow accurate models to enable queue bandwidth dynamic optimization," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, 2020, pp. 231–235.

[31] L. Tan, *Resource Allocation and Performance Optimization in Communication Networks and the Internet*. Boca Raton, FL, USA: CRC Press, 2017.

[32] A. M. Abdelmoniem and B. Bensaou, "Hysteresis-based active queue management for TCP traffic in data centers," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, May 2019, pp. 1621–1629.

[33] F. Smarra, A. D'Innocenzo, and M. D. Di Benedetto, "Fault tolerant stabilizability of MIMO multi-hop control networks," *IFAC Proc. Volumes*, vol. 45, no. 26, pp. 79–84, 2012.

[34] J. Yang and S. Ulukus, "Trading rate for balanced queue lengths for network delay minimization," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 5, pp. 988–996, May 2011.

[35] J. Zhang, F. Ren, X. Yue, R. Shu, and C. Lin, "Sharing bandwidth by allocating switch buffer in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 1, pp. 39–51, Jan. 2014.

[36] J. John and R. Balan, "Priority queuing technique promoting deadline sensitive data transfers in router based heterogeneous networks," *Int. J. Appl. Eng. Res.*, vol. 12, no. 15, pp. 4899–4903, 2017.

[37] O. Lemeshko, T. Lebedenko, O. Yeremenko, and O. Simonenko, "Mathematical model of queue management with flows aggregation and bandwidth allocation," in *Advances in Computer Science for Engineering and Education* (Advances in Intelligent Systems and Computing). Springer, 2018, pp. 165–176.

[38] M. Bahnasy, H. Elbiaze, and B. Boughzala, "Zero-queue Ethernet congestion control protocol based on available bandwidth estimation," *J. Netw. Comput. Appl.*, vol. 105, pp. 1–20, Mar. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S108480451730423X

[39] M. Kim, M. Jaseemuddin, and A. Anpalagan, "Deep reinforcement learning based active queue management for IoT networks," *J. Netw. Syst. Manag.*, vol. 29, no. 3, pp. 1–28, 2021.

[40] L. Boero, M. Cello, C. Garibotto, M. Marchese, and M. Mongelli, "BeaQoS: Load balancing and deadline management of queues in an OpenFlow SDN switch," *Comput. Netw.*, vol. 106, pp. 161–170, Sep. 2016.

[41] K. S. Umadevi, M. S. S. Pranay, and K. Rachana, "Multilevel queue scheduling in software defined networks," in *Proc. Innov. Power Adv. Comput. Technol. (i-PACT)*, 2017, pp. 1–4.

[42] C. Olariu, M. Zuber, and C. Thorpe, "Delay-based priority queueing for VoIP over software defined networks," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manag. (IM)*, May 2017, pp. 652–655.

[43] H. Ma, J. Yan, P. Georgopoulos, and B. Plattner, "Towards SDN based queuing delay estimation," *China Commun.*, vol. 13, no. 3, pp. 27–36, 2016.

[44] Y.-J. Chen, L.-C. Wang, F.-Y. Lin, and B.-S. P. Lin, "Deterministic quality of service guarantee for dynamic service chaining in software defined networking," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 4, pp. 991–1002, Dec. 2017.

[45] A. Al-Najjar, S. Layeghy, M. Portmann, and J. Indulska, "Enhancing quality of experience of VoIP traffic in SDN based end-hosts," in *Proc. 28th Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, 2018, pp. 1–8.

[46] Q.-L. Li, J.-Y. Ma, R.-N. Fan, and L. Xia, "An overview for Markov decision processes in queues and networks," in *Proc. Int. Conf. Celebrating Professor Jinhua Cao's 80th Birthday*, Oct. 2019, pp. 44–71.

[47] A. Asanjarani, Y. Nazarathy, and P. K. Pollett, "Parameter and state estimation in queues and related stochastic models: A bibliography," 2017, *arXiv:1701.08338*.

[48] K. Sood, S. Yu, and Y. Xiang, "Performance analysis of software-defined network switch using M/geo/1 model," *IEEE Commun. Lett.*, vol. 20, no. 12, pp. 2522–2525, Dec. 2016.

[49] D. Singh, B. Ng, Y.-C. Lai, Y.-D. Lin, and W. K. G. Seah, "Modelling software-defined networking: Software and hardware switches," *J. Netw. Comput. Appl.*, vol. 122, pp. 24–36, Nov. 2018.

[50] R. Schoeffauer and G. Wunder, "Model-predictive control for discrete-time queueing networks with varying topology," 2020, *arXiv:2004.01985*.

[51] P. Wang, H. Chen, X. Yang, and Y. Ma, "Design and analysis of a model predictive controller for active queue management," *ISA Trans.*, vol. 51, no. 1, pp. 120–131, 2012.

[52] K. B. Kim, "Design of feedback controls supporting TCP based on the state-space approach," *IEEE Trans. Autom. Control*, vol. 51, no. 7, pp. 1086–1099, Jul. 2006.

[53] F. Smarra, A. Jain, T. de Rubeis, D. Ambrosini, A. D'Innocenzo, and R. Mangharam, "Data-driven model predictive control using random forests for building energy optimization and climate control," *Appl. Energy*, vol. 226, pp. 1252–1272, Sep. 2018.

[54] (May 2019). *Mininet*. [Online]. Available: http://mininet.org/

[55] S. Avallone, S. Guadagno, D. Emma, A. Pescapè, and G. Ventre, "D-ITG distributed internet traffic generator," in *Proc. 1st Int. Conf. Quant. Eval. Syst. (QEST)*, Sep. 2004, pp. 316–317.

[56] A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Comput. Netw.*, vol. 56, no. 15, pp. 3531–3547, Oct. 2012.

[57] A. Botta, W. de Donato, A. Dainotti, S. Avallone, and A. Pescape, "D-ITG 2.8. 1 manual," Comput. Interact. Commun. (COMICS) Group, Dept. Elect. Eng. Inf. Technol., Univ. Naples Federico II, Naples, Italy, 2013. [Online]. Available: https://www.grid.unina.it/software/ITG/manual

[58] (May 2019). *Open vSwitch*. [Online]. Available: https://www.openvswitch.org/

[59] F. Baker, D. Black, S. Blake, and K. Nichols, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, document RFC 2474, Dec. 1998.

[60] J. Babiarz, K. Chan, and F. Baker, *Configuration Guidelines for Diffserv Service Classes*, document RFC 4594, Aug. 2006.

[61] A. M. Langellotti, S. Mastropietro, F. T. Moretti, and A. Soldati, "Notiziario tecnico telecom Italia," Telecom Italia, Rome, Italy, Tech. Rep., 2004.

[62] *Ryu.app.ofctl Rest*. [Online]. Available: https://ryu.readthedocs.io/en/latest/app/ofctl_rest.html

[63] (May 2019). *QoS Ryubook 1.0 Documentation*. [Online]. Available: https://osrg.github.io/ryu-book/en/html/rest_qos.html

[64] F. Smarra, A. Jain, R. Mangharam, and A. D'Innocenzo, "Data-driven switched affine modeling for model predictive control," in *Proc. IFAC Conf. Anal. Design Hybrid Syst. (ADHS)*, 2018, pp. 199–204.

[65] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[66] L. Breiman, *Classification and Regression Trees*. Evanston, IL, USA: Routledge, 2017.

[67] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[68] (May 2019). *UDOO ×86*. [Online]. Available: https://www.udoo.org/

[69] Cacti. *Last Software Realese 1.2.22*. Accessed: Aug. 2022. [Online]. Available: https://www.cacti.net/

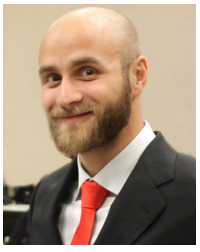
[70] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, 2018, Art. no. e00938.

[71] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," in *Proc. 12th USENIX Conf. Oper. Syst. Design Implement. (OSDI)*, 2015. [Online]. Available: https://www.tensorflow.org/

[72] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io

[73] Artificial Intelligence Techniques. (2019). *OpenNN*. [Online]. Available: https://www.opennn.net

[74] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Netw.*, vol. 6, no. 4, pp. 525–533, Jan. 1993.



ENRICO RETICCIOLI received the bachelor's degree in information engineering, the master's degree *(summa cum laude)* in telecommunication engineering, and the Ph.D. degree in information and communication technology (system engineering, telecommunications, and HW/SW platforms) from the University of L'Aquila, Italy, in 2015, 2017, and 2021, respectively. His current research interests include software defined networking (SDN) controller design based on data driven, universal software radio peripheral (USRP) programmations for telecommunication purpose, and software design of the supervisory control and data acquisition (SCADA) systems for building automation studies. He received the Certified LabVIEW Associate Developer (CLAD) Award, in 2016.

GIOVANNI DOMENICO DI GIROLAMO received the Ph.D. degree in complex systems, in 2017. From September 2015 to June 2016, he was a Visiting Student at the Department of Applied Mathematics, Université Catholique de Louvain, where studies of quantized control systems have been conducted. He is currently a Postdoctoral Researcher at the Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila. His research interests include control theory and identification, wireless networked control systems, structural health monitoring, and machine learning. His Ph.D. thesis titled: Co-design of controllers and information flows in networked control systems.

FRANCESCO SMARRA received the M.Sc. degree in control and systems engineering and the Ph.D. degree in electrical and information engineering from the University of L'Aquila, L'Aquila, Italy, in 2010 and 2014, respectively. He accomplished the International Curriculum Option of Doctoral Studies in networked, embedded, and hybrid control systems for complex, distributed and heterogeneous systems. He is currently an Assistant Professor at the Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila. He has held visiting scholar positions at UC Berkeley, from 2012 to 2013 and University of Pennsylvania, from 2014 to 2015, and visiting research position at University of Pennsylvania, from 2016 to 2017. His research interests include control theory, wireless networked control systems, building automation systems, structural health monitoring systems, and machine learning. He was a recipient of the Fondazione Filauro Award for Ph.D. students, in 2013 and the Best Application Paper Award of the European Control Conference, in 2015.

ANGELO TORZI has been the Founder and the Director of Sonicatel s.r.l., since 2010, which aims to develop innovative networks and solutions, dedicated to business customers. He has 20 years of experience in the telecommunications and information technology sector, holding various managerial positions.

FABIO GRAZIOSI (Member, IEEE) was born in L'Aquila, Italy, in 1968. He received the Laurea degree *(cum laude)* and the Ph.D. degree in electronic engineering from the University of L'Aquila, L'Aquila, in 1993 and 1997, respectively. He is currently a Full Professor in telecommunications at the Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila. He is the Coordinator of the INCIPICT Project aiming at promoting a Living Laboratory focused on the IoT and Smart City for the City of L'Aquila. He has authored over 200 papers in technical journals and conference proceedings. His current research interests include wireless communications for the IoT, with emphasis on wireless sensor networks, cognitive radio, and cooperative communications. He is a member of the Scientific Committee of the "Consorzio Nazionale Interuniversitario per le Telecomuni-cazioni" (CNIT). He is also the Chairperson of the Board of Directors of WEST Aquila s.r.l., a spin-off Research and Development company of the University of L'Aquila. He is involved in major national and European research programs in the field of wireless systems and he has been a reviewer for major technical journals and international conferences in communications. He also serves as a technical program committee (TPC) member and the session Chairperson for several international conferences in communications.

ALESSANDRO D'INNOCENZO (Member, IEEE) received the Ph.D. degree in electrical and information engineering from the University of L'Aquila, in 2007. From 2007 and 2009, he was a Postdoctoral Researcher at the University of L'Aquila. In 2008, he was a Postdoctoral Researcher at the University of Pennsylvania. He is currently an Associate Professor with the Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila. His research interests include control theory and in particular, networked control, learning in control, with applications to smart cities and communication networks automation. He was a recipient of the Best Application Paper Award of the European Control Conference, in 2005.