RoPE: An Architecture for Adaptive Data-Driven Routing Prediction at the Edge

RoPE: An Architecture for Adaptive Data-Driven Routing Prediction at the Edge

Alessio Sacco Flavio Esposito Guido Marchetto

***Abstract*—The demand of low latency applications has fostered interest in edge computing, a recent paradigm in which data is processed locally, at the edge of the network. The challenge of delivering services with low-latency and high bandwidth requirements has seen the flourishing of Software-Defined Networking (SDN) solutions that utilize ad-hoc data-driven statistical learning solutions to dynamically steer edge computing resources.**

In this paper, we propose RoPE, an architecture that adapts the routing strategy of the underlying edge network based on future available bandwidth. The bandwidth prediction method is a policy that we adjust dynamically based on the required time-to-solution and on the available data. An SDN controller keeps track of past link loads and takes a new route if the current path is predicted to be congested.

We tested RoPE on different use case applications comparing different well-known prediction policies. Our evaluation results demonstrate that our adaptive solution outperforms other ad-hoc routing solutions and edge-based applications, in turn, benefit from adaptive routing, as long as the prediction is accurate and easy to obtain.

***Index Terms*—SDN, Edge computing, Adaptive routing, Machine Learning.**

I. Introduction

Edge computing, combined with network softwarization has been a *petri dish* for new business models and applications, promising simultaneously low-latency and high-bandwidth reliable telecommunications. This paradigm has moved computation closer to the network traffic source, reducing delays with respect to standard cloud computing applications [1]–[4].

Delivering such promises is, however, a challenge, especially when the underlying infrastructure is unstable and applications impose tight constraints. Solutions for real-time communications have been proposed when the application is bound to video streaming [5]–[9]. Many of them are based on sound design and target bit rate adaptation. Aside from ignoring other end-to-end performance improvement techniques such as traffic compression, these solutions perform poorly within edge computing use cases, where the underlying network needs also to be optimized, in response to offloading requests [10]–[12].

Other solutions, *e.g.*, [13] seek help from network traces to forecast future demands. Most of these solutions, however, train their learning system on specific datasets, without the ability to adapt. While complicated machine learning techniques such as transfer learning exist [14], such techniques could be applied to overcome the dataset-tailored limitation. In this paper, we take a more humble approach and we show its effectiveness. In particular, we introduce RoPE[1], a Software-Defined Networking (SDN)-based architecture whose goal is to select the best (physical or virtual) route by applying the most appropriate bandwidth prediction algorithm, chosen adaptively, on the basis of the amount of data collected and the response time deadline. RoPE leverages the availability of multiple paths and relies on the idea that the bottleneck for delay-sensitive applications is at the edge [15], [16].

[1]RoPE stands for Routing Prediction at the Edge.

Our design is based on the observation that, in recent years, the field of prediction has achieved excellent results when enough data are available. When insufficient data are available instead, other classes of prediction algorithms may be a better fit. In this context, many forecast-based or data-driven solutions have been proposed [14]. The question we propose to answer instead in this paper is: *which bandwidth prediction algorithm works best, based on the variance of our network measurements and on application constraints?*

To address this question, we prototype and evaluate RoPE with both numerical, event-driven simulations, and with scalability tests over the large-scale GENI testbed. In particular, we make the following contributions.

Our contribution. We design and implement a novel architecture that integrates QoE estimation and bandwidth prediction directly into an edge-based application. The prediction phase is used for selecting the best routes on the basis of global traffic condition information gathered from an SDN controller. Hence, we defined a new strategy for the route selection while the prediction continues during system operation, with consequent possible traffic re-routing.

To adapt to different edge-based applications and evaluate its performance, we define a new cost function that embraces the most common evaluation parameters.The collection of our results evaluating three separate uses cases are a mixture of expected and surprising results.

The structure of the paper is as follows. Section II summarizes the related work. Section III introduces in which use cases the solution can be used, while Section IV presents the system in which the routing algorithm is applied, as well as the framework and the overall procedure. In Section V a brief explanation about prediction algorithms is provided and Section VI shows the quality and the differences among these algorithms. Finally, Section VII demonstrates the benefits of this new approach and Session VIII concludes the paper.

II. Related Work

Our approach is based on the prediction of traffic conditions to modify routing for edge-based applications. In this section, we analyze the literature related to the main components of the solution: (i) the recent prediction algorithms for networking,

and (ii) the existing routing solutions that rely on machine learning methods to improve traditional strategies.

A. Network Traffic Prediction

The prediction of traffic conditions is crucial in network operations and management for today's increasingly complex and diverse networks. It entails forecasting future traffic and traditionally it has been addressed via Time Series (TS) algorithms. The main goal in TS is to construct a regression model capable of drawing an accurate correlation between future traffic volume and previously observed traffic volumes. Existing TS models can be broadly decomposed into statistical analysis models and supervised ML models. Statistical analysis models are typically built upon the generalized Autoregressive Integrated Moving Average (ARIMA) method, while the majority of learning for traffic prediction is achieved via supervised Neural Networks (NNs). However, with the rapid growth of networks and the increasing complexity of network traffic, traditional statistical models are seemingly compromised, giving rise to more advanced ML models [14]. More recently, efforts have been made to reduce overhead or improve accuracy in traffic prediction by employing features from flows, other than traffic volume. Prior work focused on NNs and showed how this approach outperforms TS [17]. However, the use of NNs implies an offline training phase and a huge quantity of training data, that is unfeasible for some applications [18]. In our scenario we don't have such a quantity, therefore we focus on lighter approaches, that enable an online training phase. These models are then compared against Machine Learning methods where the training is performed offline. Furthermore, for edge-based applications, there are no databases available online as for traffic traces provided by ISPs or inter and intra DCs [19].

For this reason, in our work we focus on other ML algorithms that also do not necessitate a long training phase. Many techniques have been developed to measure path properties as summarized by CAIDA [20]. In particular, several studies [21]–[23] focused on the measurement of the available bandwidth, needed for data collection in our predictor. By available bandwidth, we mean the minimum unused capacity on a given end-to-end path. These measurements are usually collected with probe packets. In this work, we do not actively probe but we rely on packets sent from switches to the controller. In this way packets used for the collection of network statistics do not affect the user data, since the communication with the controller is separated from the data plane [24], [25].

Finally, machine learning techniques have been widely applied to network measurement. For example, there are applications in the network intrusion detection field (e.g., [26]) and for round-trip time prediction [27]. In contrast to NNs-based algorithms, Support Vector Machine (SVM) has low computational overhead and is more robust to parameter variations, *e.g.*, time scale, number of samples. However, this approach is usually applied to classification rather than regression. Bermolen *et al.* [28] applied SVR (the regression variant of SVM) for link load forecasting. Furthermore, He *et al.* [29] extensively studied history-based predictors using three different time series forecasts. Other approaches for TCP throughput prediction employ "bandwidth probes", small TCP file transfers, *e.g.*, 64kB, to collect the measures [30], [31].

B. Adaptive Routing and Traffic Engineering

Even though much work has been conducted to improve the quality of prediction over network traffic, only a few solutions exploited these results to develop new routing strategies [32], [33].

Instead, other prediction-driven routing approaches have been based on Reinforcement Learning (RL), with the aim of coping and scaling to complex and dynamic network topologies [34], [35]. Even though RL would be a viable solution, we used a time-series approach as it offers the possibility of predicting a specific parameter. Such a parameter can then in turn be used to assess if a given traffic flow fits a physical path. If the flow does not fit the path, a better route is chosen looking at other available paths.

The same problem can be clearly addressed by means of traffic engineering solutions, *e.g.*, [36]–[38]. In particular, COYOTE [38] aims to minimize link over-utilization by engineering the traffic generated with optimal traffic splitting ratios. Given the limited knowledge of traffic demands, this method strategically advertises fake links and nodes to adjust the splitting ratios resulting from the traditional ECMP mechanism. We share with this solution the idea of adapting the routing to address a link utilization problem; however, our focus is to better support for edge-based applications without reserving resources for tasks that could be rarely executed.

III. Motivating Applications

Edge-based applications have evolved in the last decade because of a considerable demand [39]. Many applications have strict requirements to satisfy, *e.g.*, very low latency and high throughput. In the following subsections, we analyze three applications that we consider as use cases for our study.

A. Very Latency-sensitive Applications: Tactile Internet

The Tactile Internet is the evolution of the mobile Internet and enables real-time control of the Internet of Things (IoT). It adds a new dimension to human-to-machine interaction by enabling tactile and sensations, and at the same time revolutionizes the interaction of machines. The Tactile Internet enables haptic interaction with visual feedback. The term haptic relates to the sense of touch, in particular, the perception and manipulation of objects using touch. The visual feedback will encompass not just audiovisual interaction, but also robotic systems that can be controlled in real-time as well as actuating robots, *i.e.*, those that can activate a motion.

Nowadays, data rates increased in the orders of magnitude, as well as the data capacity [40], but there is another frontier to be considered: the reduction in the end-to-end latency of interaction has not dropped below the requirement for telephony. Long-term evolution (LTE) achieves a typical end-to-end latency close to 100ms [41], exceeding the order of 1-ms requirement needed to enable Tactile Internet applications [40]. At the same time, fifth generation (5G) mobile communications systems underpin this emerging Internet at the wireless edge [42]. A recent trend is the use of Mobile

Edge Computing (MEC) as a solution to reduce the delay and provide a way for offloading computation from the cellular network [43]. However, the latency reduction is still an open problem due to an intrinsic lack of the available infrastructures. The SDN paradigm is shown to be helpful for these applications [44], but real support for very low latency communications is an urgent need to enable the still unexpressed haptic applications.

B. Telepathology at the Edge.

The field of medical pathology is concerned with the causal study of disease, whether caused by pathogens or non-infectious physiological disorders. A significant part of the job of pathologists is characterized by visualizing histological images via a multi-lens microscope. Often they analyze histological images on a glass slide when the patient is still under a tumor removal surgery. In such situations, a quick and correct pathology assessment is crucial as it defines vital next steps for the surgeon team and the right follow-up treatment for the patient. In the vast majority of non-trivial pathology cases, to minimize the time to response to the surgeon team and the probability of incorrect assessments, pathologists ask for second opinions to nearby experts (if available) by physically carrying privacy protected glass specimens. When not enough local experts are available, a *telepathology* system can be used to transmit high-resolution images of specimens to remote doctors.

Telepathology solutions can be used not only to connect rare experts with patients, but also for the rapid diagnosis of standard cases in locations that have patients without having schools of medicine. Telepathology is often enclosed in the telemedicine field, but it differs both in the subject and the aim of such practice. This difference leads to different requirements that the underlying network has to guarantee [45].

In particular, current telepathology solutions are limited by the technology, the scale, and the (best-effort) performance of the underlying telecommunication media on which they rely on, *i.e.*, the Internet or, at best, a virtual private network for in-hospital offline, *i.e.*, non-real-time, consultations.

A Telepathology or more generally, a Telemedicine session transmits delay and bandwidth sensitive data to be processed and shared with a remote medical doctor. For this reason, a proper edge computing system can be implemented to partially or fully offload processes at the edge of the network [2].

C. Disaster-response

Providing technologies in response to a natural or man-made disaster is challenging, as the assumptions typically made for traditional infrastructure may fail given the damage made by the disaster. Additionally, mobile applications that serve the needs of disaster incident response generate large datasets and demand large computational resources. These datasets are usually collected in real-time at the disaster scene using different IoT devices. Examples of such devices are wearable heads-up devices, Unmanned Aerial Vehicles equipped with sensors, cameras, or smartphones [46], [47]. For example, such devices might be used for real-time video conferencing with the incident commander featuring face recognition of disaster victims [48], or to detect children in an attempt to reunite them with their guardians [49], whereas virtual beacons can be mainly used to track their location.

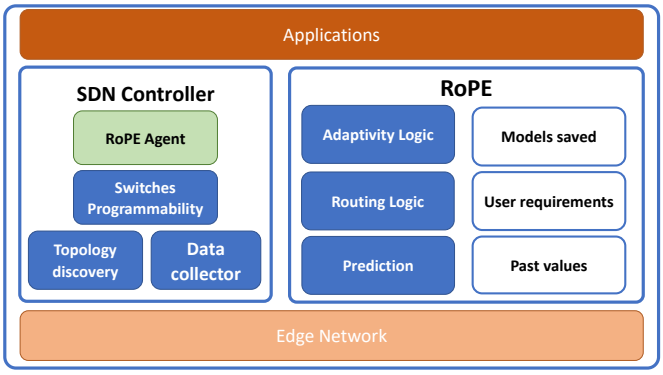


Fig. 1: Architecture and main RoPE's functionalities.

To enable immediate feedback to first responders, crucial for survivors' rescue, IoT devices today could benefit from the mobile edge computing paradigm [50]. In particular, one of the most important mechanisms in edge computing is cyber foraging: processes from mobile resources delegate computations or code to execute to servers within the edge computing infrastructure [51]. A particular case of cyber foraging is also known as offloading, where the cyber foraging mechanism is orchestrated by mobile devices.

To cope with the potential loss of infrastructure in a disaster scene, an edge network needs to be operational for transferring media-rich visual information from the disaster scene as quickly as possible to the edge cloud gateway. Such (visual) data can be used, *e.g.*, within a medical application context to transfer high-definition video streams generated by paramedics' wearable heads-up display devices from the disaster triage scene to a dashboard located at the edge cloud, or by a first responder for a live remote medical consultation. The incident response and resource allocation decision making e.g., ambulance routing to the scene or medical supply replenishment tracking requires significant computational resources that can be augmented on demand by a core cloud cluster.

For such scenarios to be operational, traffic needs to be handled dynamically and with low-latency constraints. Hence routing is a crucial infrastructure management orchestration mechanism. Although geographic routing-based approaches are generally suitable for these applications; there is a lack of providing sustainable high-speed data delivery to the edge cloud gateway [52].

IV. RoPE Architecture and System Design

In this section, we describe the design of our *ROuting Prediction at the Edge* architecture, or *RoPE*. We begin with an overview of the principles, and we detail each component of the architecture in the subsequent subsections.

A. RoPE Overview

Our proposal is to use bandwidth prediction on links to drive routing operations so that the best available path is selected. Given a large number of available prediction algorithms and the differences in requirements to satisfy each application,

we also introduce a cost function that captures the policy programmability of the proper algorithm for each specific context. The design goal of such policy knobs is to extract the invariances in the routing prediction mechanism. Network management application programmers then may tune this utility based on their needs and constraints.

Our architecture implementation includes a Ryu SDN controller that collects data from the switches and communicates them to the RoPE agent (Figure 1) running on it. This component allows the necessary information sharing between the controller and RoPE. In RoPE, the most important component is responsible for predicting on the basis of the data received and prior information. The data-driven engine selects the best path combining user requirements and the future available bandwidth on a link. To select the best path, knowledge about the topology of the network is necessary, and this information is obtained and transmitted by the controller. The routing process combines the output of the prediction with the topology information and changes the flow rules of the switches to select the path consistently among all the devices.

RoPE saves the collected data as recent history, which is in turn used by the prediction algorithms. Notice that not all the algorithms need online training (see Section V). For some algorithms, the training phase must be performed offline because it requires a long time, as illustrated in Section VII-A. For these algorithms, the SDN Controller can make use of the saved model to predict the next bandwidth value. In essence, the prediction might be based on models saved and past values, as shown in Figure 1. The selection of all the parameters is based on the data analysis performed beforehand and described in the following sections.

B. Measurements Collection

Each managed switch is connected to the Ryu controller, which periodically collects information on their state. In particular, we collect network state statistics of ports (incoming or outgoing packets), flows, and the switch connectivity status.

Since paths do not change very frequently, it is unnecessary to acquire statistics from switches with very high granularity. In our implementation, we use a collection period of 5-seconds, as in [5]. In Section VII-D we motivate in details this parameter choice with our analysis. In the rest of the paper, we refer to such interval as an *epoch*.

Data acquired are grouped per switch ID and in chronological order. This is implemented on the controller by logging the received information in a file for every switch. Each row in this file corresponds to an observation per epoch and is formatted as follow:

$$[timestamp, bandwidth, bytes, packets, packets\_port],$$

where $timestamp$ denotes the time at which the record is obtained, $bandwidth$ is the the measured bandwidth, $bytes$ refers to the number of bytes sent and received by the switch, $packets$ expresses the total number of packets sent and received by the switch, and, lastly $packets\_port$ indicates the amount of packets sent and received in the selected port. Note how the timestamp is essential to apply TS analysis, while it is not used by ML algorithms. RoPE uses the statistics collected to fit the model. With a period r of 20 s (selected to avoid network instability, see Section VII-D) we predict the future available bandwidth and decide when to steer the route.

Algorithm 1 Prediction-based routing

```
Let t be the epoch, and r the prediction period
Let A and B be the target source and destination
P ← all the paths between A and B
P_s ← the best s paths in P
U ← best path
for every epoch t do
    Monitor the path in P_s
    if r has elapsed since last prediction then
        FL_s ← future predicted load on the s paths in P_s
        FL_U ← future predicted load on U
        if FL_U > Threshold then
            U ← P[min(FL_s)].
            close;
```

Overall Procedure. The objective of the algorithm is to optimize the available bandwidth between the source *A* and the destination *B*, which affects the application transmission time. In the telemedicine example described before, *A* is the Plugin connected to the microscope, while *B* is the edge server. The controller detects the best s paths for the pair (A, B) and stores them into the set P_s. The parameter s is used to avoid looking for all the paths if this number is significant. Every epoch, the controller obtains the statistics and saves them for the prediction, which occurs every period r. In this phase, we estimate the future value for the paths in P_s, and the path whose prediction is the minimum, *i.e.*, "argmin", is set as the default one.

C. Cost Function

RoPE predicts the bandwidth on links and selects the best path on the basis of this value. However, different applications have different requirements in terms of throughput, latency, jitter, and different prediction algorithms may have different effects on these parameters.

To evaluate the fitness of such an algorithm to the use case, we define a cost function $C_{K,I}(D)$ that takes into account the above aspects of communication. While the metric is inspired by similar studies [53] in our case we are not limited to video streaming scenarios. The cost function $C_{K,I}(D)$ of a sent file which requires D bytes, I packets and K time intervals, is made up of the following terms:

1) *Average Throughput*: the average throughput per time interval k: $\frac{1}{K}\sum_{k=1}^{K} T_k$, where T_k denotes the throughput at interval k
2) *Average latency*: the average latency per packet i:$\frac{1}{I}\sum_{i=1}^{I} L_i$, where L_i is the latency for packet i
3) *Average jitter*: the average jitter between two consecutive packets: $\frac{1}{I-1}\sum_{i=2}^{I}|L_i - L_{i-1}| = \frac{1}{I-1}\sum_{i=2}^{I} J_i$, where J_i indicates the jitter for packet i
4) *Average jitter variation*: the average difference of jitter among two consecutive jitter measurements $\frac{1}{I-2}\sum_{i=3}^{I}|J_i - J_{i-1}| = \frac{1}{I-2}\sum_{i=3}^{I}\Delta J_i$ where ΔJ_i refers to the jitter variation for packet i

Notice that, besides the standard performance metrics of throughput, latency, and jitter, it is worth also considering the jitter variation since for interactive systems, such as Tactile and Telepathology, this element affects the user experience, as demonstrated in [54], [55]. Users and application programmers may have different preferences on which of the four components is more important, so we define a tunable objective function as a weighted sum of the aforementioned components:

$$C_{K,I}(D) = \alpha D \frac{K}{\sum_{k=1}^{K} T_k} + \mu \frac{1}{I} \sum_{i=1}^{I} L_i + \\ + \lambda \frac{1}{I-1} \sum_{i=2}^{I} J_i + \gamma \frac{1}{I-2} \sum_{i=3}^{I} \Delta J_i \quad (1)$$

Here $\alpha, \mu, \lambda, \gamma$ are non-negative weighting parameters corresponding to average throughput, average latency, average jitter and average jitter variation respectively. A relatively small α indicates that the user is not particularly concerned about a very high bitrate; the large γ is, the more effort is made to achieve smoother changes of video quality. A large μ, relatively to the other parameters, indicates that a user is deeply concerned about low latency communication.

In summary, this definition of $C_{K,I}(D)$ is quite general as it allows us to model varying user preferences on different contributing factors. The goal of our routing strategy is to minimize (1) in order to guarantee the optimal user experience. In fact, a higher throughput, along with lower values of latency and jitter, leads to a lower value for the function. Therefore, we need to select the proper prediction method in order to obtain the best routing strategy that minimizes (1).

V. Prediction Algorithms Analysis

The task of bandwidth prediction can be formulated as a regression problem, *i.e.*, predicting a real-valued number based on single or multiple real-valued input features. For the sake of clarity we classify the applied algorithms in 2 categories, *(i)* Time-Series (TS) algorithms, *(ii)* Machine Learning (ML) algorithms. The following subsections reflect this classification and each one describes in-depth the structure of our algorithms.

The idea is to predict the bandwidth, in such a way the controller can check whether the desired application fits the network load. For instance, if the application is sending a video streaming of 300kb/s and the predicted available bandwidth of the current path is 500kb/s, this means the path complies with the requirements. If the available bandwidth is 200kb/s, the controller enforces a new path.

A. Time-Series Models

These solutions are based on traditional regression algorithms that predict the future values using the history and the evolution of such value in the past. The history used is made up of past values associated with the timestamp. The presence of the tuple $< timestamp, value >$ leads to the name Time-Series.

Simple Exponential Smoothing. Simple Exponential Smoothing (SES) is a good choice for data with no clear trend or seasonality. Let y_t be the bandwidth on a link at time t. We compute a k-steps ahead prediction. Formally, we forecast the value of the bandwidth at time $t+k$, y_{t+k}, where k is also called horizon.

$$y_{t+k} = \alpha y_t + \alpha(1-\alpha) y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + ..., \quad (2)$$

where α is the smoothing parameter for $0 \leq \alpha \leq 1$. If α is large (close to 1), more weight is given to more recent observations. The quantity y_{t+k} represents the predicted value and is used to decide whether or not a congestion will occur.

Holt-Winters. The prediction is composed of three submodels that fit a time series: an average value, a slope (or trend) over time and a cyclical repeating pattern (seasonality) [56]. These three aspects of the time series behavior are expressed as three types of exponential smoothing. The model requires several parameters: one for each smoothing (α, β, γ), the length of a prediction season, and the number of periods in a season. Here below we report how the Holt-Winters seasonal method includes the forecast equation and three smoothing equations: one for the level L_t, one for the trend b_t and one for the seasonal component denoted by S_t, with smoothing parameters α, β and γ:

$$\begin{aligned} level\ L_t = &\quad \alpha(y_t - S_{t-s}) + (1-\alpha)(L_{t-1} + b_{t-1}), \\ trend\ b_t = &\quad \beta(L_t - LL_{t-1}) + (1-\beta) b_{t-1}, \\ seasonal\ S_t = &\quad \gamma(y_t - L_t) + (1-\gamma) S_{t-s}, \\ forecast\ y_{t+k} = &\quad Lt + kb_t + S_{t+k-s}, \end{aligned}$$

where s is the length of the seasonal cycle, for $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$ and $0 \leq \gamma \leq 1$.

ARIMA. ARIMA is a class of models typically used for analyzing and forecasting time series (*e.g.*, financial market data). A standard notation for this method is ARIMA(p, d, q), where the parameters account for seasonality, trend, and noise in datasets. In particular, p captures the auto-regressive component *i.e.*, the number of lag observations included in the model, also called the "lag order"; d captures the integrated part of the model, it is the number of times that the raw observations are differenced, also called the degree of differencing; q captures the moving average part of the model and represents the size of the moving average window, also called the order of moving average. The ARIMA overall model is given by the following equation:

$$\left(1 - \sum_{i=1}^{p} \alpha_i L^i\right)(1-L)^d y_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \epsilon_t, \quad (3)$$

where L is the lag operator — the number of past samples considered during the prediction — and α_i are the parameters of the autoregressive part of the model; the θ_i are the parameters of the moving average while ϵ_t are error terms. Such error terms ϵ_t are generally assumed to be independent and identically distributed (i.i.d.) variables sampled from a normal distribution with zero mean, which is what we did.

SARIMA. To deal with seasonal effects, we make use of the seasonal ARIMA (SARIMA), which is denoted as ARIMA$(p, d, q)(P, D, Q)s$. Here, (p, d, q) are the non-seasonal parameters described above, while (P, D, Q) follow

the same definition but correspond to the seasonal components of the time series. The term s is the periodicity of the model (4 for quarterly periods, 12 for yearly periods, etc.).

The Ryu controller is in charge of collecting all bandwidth values and save them in a time series $Y = \{y_t, y_{t-1}, ..\}$. The sequence is then used to fit the model and find the aforementioned parameters. Once the model is built, it is used to forecast the y_{t+k} value, which is then used to avoid congested paths in a telepathology session.

B. Machine Learning Algorithms

Machine Learning has received great attention in recent years, due to the ease of use and the wide range of applications that can benefits. In this section we define a model for the most popular algorithms, providing a brief explanation of the advantages and disadvantages of applying for each of them. In our model the set of features used is represented by [timestamp, bandwidth, bytes, packets, packets_port], however, for ML methods only a subset is considered:

a) Δ *Packets:* the number of packets received and transmitted by the switch in the time interval;
b) Δ *Bytes:* the number of bytes received and transmitted by the switch in the time interval;
c) Δ *Packets port:* the number of packets received and transmitted by the switch on a certain port in the time interval.

Our problem lies in the Regression procedure since the aim is to predict a continuous value, as opposed to other well-known problems such as classification and clustering. A real number is more effective than a class value as in the Classification problem because it can be used to check if a streaming video will be delayed or not, as described in Section VII. By computing the available bandwidth on a path, we are able to verify whether the bit-rate of communication fits the path or not, and in case move to another path. Hence, the output variable is the bandwidth of the links connected to the switch. The predicted value is the same as the TS models, while in ML models the input set is based on more features than just the past bandwidth.

Linear. The simplest machine learning model is to build a linear regression model, where there is a linear relationship between the dependent (y) variable and the set of independent (x) variables.

Polynomial. Polynomial regression is a special case of linear regression. But in this case, higher order powers (2nd, 3rd or higher) of an independent variable are included.

Support Vector Regression. Support Vector Machines (SVMs) are supervised learning models [57], that aim to analyze data and recognize patterns, used for classification tasks. Support Vector Regression (SVR), is the regression version of the popular SVM and a state-of-the-art machine learning tool for multivariate regression.

Gradient Boosting Regression. Gradient boosting is a machine learning technique used both for regression and classification problems. Like other boosting methods, it builds the model in a stage-wise fashion, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. The intuition behind the gradient boosting algorithm is to repetitively leverage the patterns in residuals and strengthen a model with weak predictions and improve it. Once a stage that do not have any pattern that could be modeled is reached, residuals modeling can be stopped (otherwise it might lead to overfitting). In other words, for Gradient Boosting Regression (GBR) a regression tree is fit on the negative gradient of the given loss function.

Partial Least Squares Regression. Partial least squares regression (PLSR) is a statistical method similar to other regressors; instead of finding hyperplanes of maximum variance between the dependent and independent variables, it finds a linear regression model by projecting the predicted variables and the input variables to a new space [58]. PLSR is used to find the fundamental relations between the two matrices X and Y, i.e. a latent variable approach to model the covariance structures in these two spaces. PLSR is particularly suited when there is multicollinearity among X values. Conversely, standard regression will fail in these cases (unless it is regularized).

Decision Tree Regression. A decision tree has a flow-chart-like structure, where each internal (non-leaf) node denotes a test on an attribute. Each branch represents the outcome of a test, and each leaf node holds a class label. The topmost node in a tree is the root node. The general approach of deriving predictions from a few simple if-then conditions can be applied to regression problems as well. Unlike linear models, Decision Tree Regression (DTR) is able to capture non-linear interaction between the features and the output value [59].

Random Forest Regression. The random forest model for regression (RFR) is a type of additive model that predicts by combining decisions from a sequence of base models. More formally this class of models can be written as:

$$g(x) = f_0(x) + f_1(x) + f_2(x) + ...,$$

where the final model g is the sum of simple base models f_i. Here, each base classifier f_i is a simple decision tree. This broad technique of using multiple models to obtain better predictive performance is also known as model ensembling. In RFR, all the base models are constructed independently using a different subsample of the data.

As a matter of fact, classical and ML methods are not that different from each other but distinguished by whether the models are more simple and interpretable or more complex and flexible. Hence, classical statistical algorithms tend to be much quicker and easier-to-use.

VI. Prediction Algorithms Evaluation

The algorithms presented in this section aim at predicting the future available bandwidth on a single path. The path consists of a certain number of link, the assumption is that the SDN controller knows the topology of the network. Nowadays many SDN controllers, e.g., Onos, Ryu, OpenDayLight, can obtain a logical view of the network topology. In our testbed Ryu is chosen as SDN controller technology due to its usability and a lighter approach as a python framework for SDN application development: thus, a faster response on flow installation was expected, as confirmed in previous work [60]. In addition,

since it is developed in Python, it has many predictors and machine learning libraries readily available.

This section exposes the logic of the methods and the errors in the prediction. Collected data are split into three sets: Training set, Validation set, and Test Set. Training Set is used to decide the parameters of the algorithm and Validation Set to compare the performance of a single family of algorithms with different settings. Finally, we use the Test Set to asses the quality of implemented algorithms.

In this section, the algorithms are compared on the basis of the accuracy of predicting. Even though ML algorithms rely on features to predict, while TS algorithms on history, we can compare the quality using standard error measures. We compute the Mean Absolute Percentage Error (MAPE) which is given by:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n} 100 \times \left|\frac{y_t - \overline{y_t}}{y_t}\right|, \tag{4}$$

where y_t and $\overline{y_t}$ are the real and the predicted observations.

Furthermore, for each model we compute the Root Mean Square Error (RMSE) and the Maximum Prediction Error (MAXE) to obtain information about the mean and the maximum error of the prediction. A direct comparison of the benefits for the user by applying each of these algorithms is performed in Sec.VII, where we compute the cost function defined (Eq. 1). In this section, a comparison among the algorithms on the basis of the standard errors is shown.

A. Data set

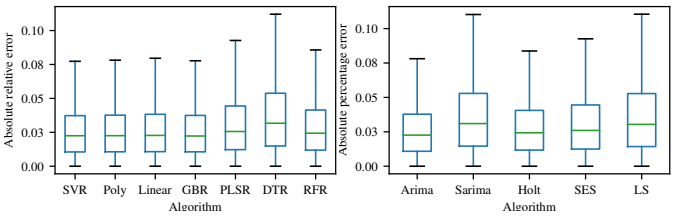
The data used in this section are collected via the Mininet network emulator. In particular, a communication between a device and a server occurs in the emulated environment to reproduce the critical traffic in the edge network. This is a video streaming application, where a live video is sent via ffmpeg [61] from a client to a server at the edge. To realistically represent the emulated loads over physical links, we set our parameters using real publicly available Internet traces [62].

We collected a dataset made of more than $50,000$ historical samples. We then split it into training (80%), validation (10%) and test (10%) set, and the error is computed on the test only. The bottom line, however, is that we cannot know for sure which approach results in the best QoE and so it becomes necessary to compare model performance and extensively study methods properties. The framework choose which model to use in light of these findings.

B. Pre-processing of Collected Data

The first thing to analyze is the stationarity of the time series. For classical forecasting methods, we want to clear trend and seasonality in the time series, by modeling these components and removing them from observations. When using ML algorithms, the stationarity test is another source of information. Hence, it can be used to select and engineer the feature in a suitable way that copes with the non-stationarity of the data.

We performed the Augmented Dickey-Fuller (ADF) test to check and confirm evidence that the time series is stationary or non-stationary. The null hypothesis of such a test is that the time series can be represented by a unit root, that it is not stationary. The alternate hypothesis is that the time series is stationary. We use the p-value from the test to establish whether the series is stationary. A p-value below a threshold (such as 5% or 1%) suggests we reject the null hypothesis and the series is stationary, meaning it does not have some time structure.



(a) (b)

Fig. 2: Error box for Time Series and Machine learning algorithms. The results are comparable meaning the quality of TS methods.

TABLE I: ADF Test to evaluate the stationarity of a time series.

	Test Statistics	p-Value	1% Critical Value	5% Critical Value	10% Critical Value
Z(t)	-6.771	0.001	-3.433	-2.863	-2.567

Table I shows test statistic value of -6. The more negative this statistic, the more likely we are to reject the null hypothesis. This value is less than the value of -3.433 at 1%. This suggests that we can reject the null hypothesis with a significance level of less than 1%. A p-value ≤ 0.05 leads to rejecting the null hypothesis (H0) and the data is stationary; vice versa the data is non-stationary. The autocorrelation analysis is useful to understand how many lags consider in the model. If the data show low correlation or no correlation, then can be hard to predict the target values through a time series problem.

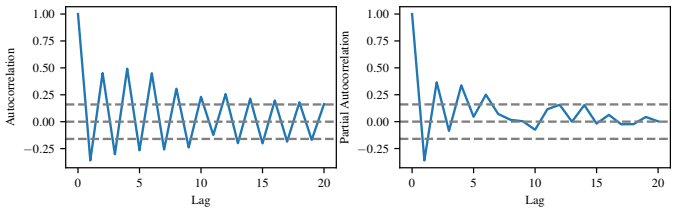
We can confer the stationarity of our data set according to our p-value. This confirms that the data do not have a trend or seasonal effects, and can be easier to model. TS methods assume or require the time series to be stationary to be effective, and results in the next corroborate quality of TS models.

C. Algorithms Analysis

We implemented the algorithms exposed in the previous section (§ V) and assess the performance for each one of them. A good predictor should at least outperform a simple algorithm in which the next value is a replica of the Last Sample (LS). This is not considered as a statistical algorithm due to the simplicity of the method, but it is a recommended baseline to compare the quality of the implemented method.

We investigate the error of the prediction of the mentioned algorithms. The results show that the error of the best TS algorithm (Holt) is comparable to the best of ML algorithms (GBR), as shown in Figure 2.

Observation 1. *Often simple models (e.g., those looking at recent epochs) are enough to achieve very low bandwidth prediction errors.*



(a) (b)

Fig. 3: (a) Autocorellation function (ACF) and (b) Partial autocorellation function (PACF) of observed data, used for tuning (S)ARIMA's parameters p and q.

TABLE II: Time-Series results when the function is fitted for each new observation.

Algorithm	MAPE	RMSE	MAXE
SES	3.12	36.45	678.69
Holt-Winters	2.87	33.17	110.41
ARIMA	2.67	30.73	597.05
SARIMA	3.70	42.83	626.22
LS	3.69	43.39	937.84

Next, we compare the most popular algorithms in the Machine Learning field, where all experiments are performed using Python implementations of the presented algorithms [63]. In addition, regarding the forecasting horizon, every model has been designed for forecasting with this horizon, since the most common usage scenario is the one-step-ahead prediction.

We define the parameters grid for each method to be searched. At the end of the process, the algorithm is tuned using the optimal set of parameters returned by this optimization process. For RFR, we define the number of estimators = [10, 50, 70, 100, 200] and random state = [0, 1, 2, None]. The same set of random states is used for DTR as well. Regarding PLSR, the number of components is set to 1, after a study performed on [0, 1, 2, 5] set. For the Polynomial model, the degree refers to the maximum exponents in the function, and we evaluated all the numbers between 2 and 7. The SVR algorithm has more parameters to be set, and we chose cost between 0.7 and 1.0, and epsilon = [0.01, 0.1, 0.5, 1.0]. The kernel value is a string, evaluated among = [rbf, poly, linear]. Finally,for GBR we set the n_estimators the same as for RFR, and learning_rate = [0.01, 0.05, 0.1, 0.5] and max_depth = [2, 3, 4, 5].

To choose the most suitable parameter combination for each method, we perform an initial study of the performance on a validation set. For each method, the parameter combination yielding the higher accuracy is chosen. The resulting parameters for ML algorithms are summarized in Table III.

Furthermore, the same process is applied to all TS methods with all parameter combinations defined in the parameter grids to train the time series. In particular, to choose the parameters for ARIMA and SARIMA, the ACF and PACF plots were used to study the parameters (Figure 3). In particular, ACF is used to determine q while PACF for p in the Equation 3. The result obtained after the grid search was compared to the statistical finding to assess the accuracy.

The Autocorrelation Function (ACF) is a measure of the correlation between the time series with a lagged version of itself. The Partial Autocorrelation Function (PACF) instead

TABLE III: Hyperparameters set in our methods.

Method	Hyperparameters
Linear	–
SVR	cost=1.0, kernel=rbf, epsilon=0.1
Polynomial	degree=4
GBR	n_estimators=500, max_depth=4, learning_rate=0.01
PLSR	n_components=1
DTR	random_state = 1
RFR	n_estimators=70, random_state=2

measures the correlation between the time series with a lagged version of itself but after eliminating the variations already explained by the intervening comparisons. These can be used to determine the p and q values as follows: p is the lag value where the Partial Autocorrelation Function (PACF) chart crosses the upper confidence interval for the first time [64], in our case $p = 1$. q instead is the lag value at which the Autocorrelation Function (ACF) chart crosses the upper confidence interval for the first time, in this case $q = 1$.

TABLE IV: Comparison of error for ML algorithms.

Predictor	MAPE	RMSE	MAXE
LINEAR	2.70	31.15	599.55
POLYNOMIAL	2.66	30.96	590.51
SVR	2.65	30.54	585.61
GBR	2.66	30.68	586.98
PLSR	3.14	37.40	885.59
DTR	3.36	41.16	539.34
RFR	2.91	33.50	580.91

To choose the best methods to address the user specification, the framework relies on the data shown in Table IV and Table II. The tables summarize the main details about errors and performance. MAPE is used to select the best algorithms, while MAXE to compare the maximum error, useful to understand the routing achievements in Section VII. Algorithms like Holt-Winters and DTR do not have the lowest error (MAPE and RMSE) but have a low MAXE. This means they are on average correct and are not far off the real value, even though the predicted value is not too close to the actual one. Routing based on this class of algorithms can achieve excellent results because they can reduce the number of false positive (wrong peak), but it can be hard to detect a true positive (real peak).

In addition to tabular data, we present scatter plots to provide insight into how median and maximum error reflect into predicted values. Fig. 4 compares the actual and predicted bandwidth value using different prediction methods. A point on the diagonal represents perfect prediction; the farther a point is from the diagonal, the greater the error. We can see that for GBR and SVR the points are limited in a closed area near the diagonal. We can notice as there are two main groups of algorithms: (i) algorithms whose prediction is stable and close to a mean value, *e.g.*, GBR, SVR, ARIMA, (ii) algorithms whose prediction is more spread and farther from the real one, *e.g.*, DTR, SES, RFR. This is related to the standard deviation of the error, it means that when the prediction is wrong, the error could be too high leading to an inappropriate conclusion. On the other hand, the prediction of the group is always close to the real value, so even though the value is not exact, the finding is likely more accurate.

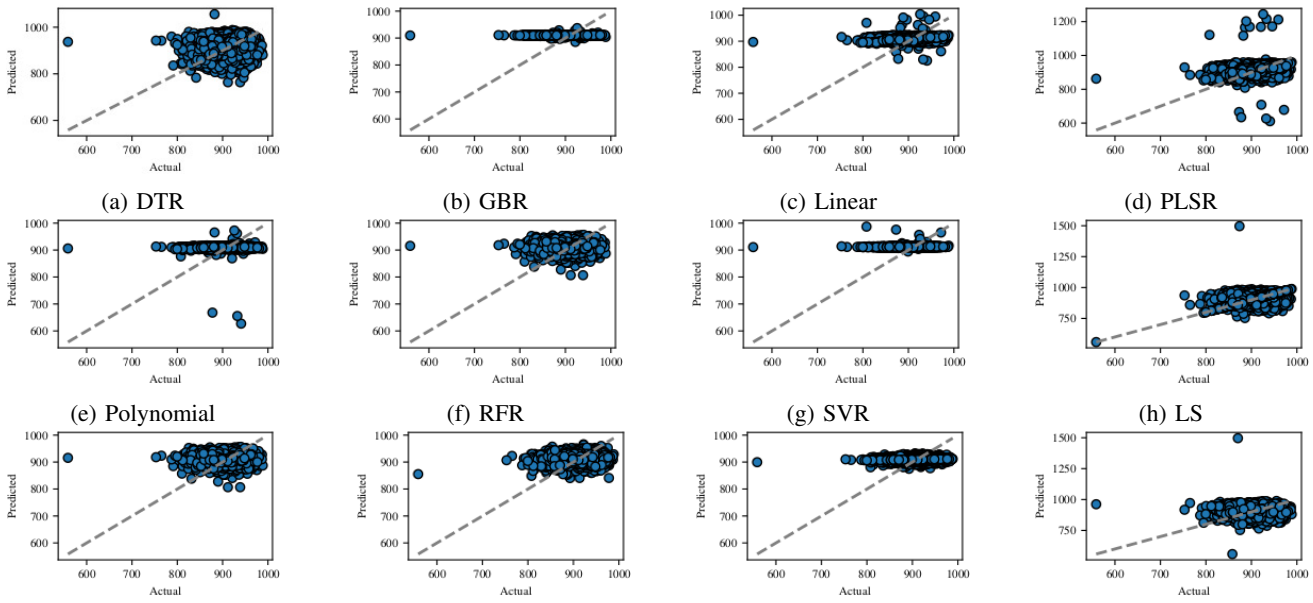


Fig. 4: Comparison of actual value (x-axis) and predicted value (y-axis). All the algorithms are tested on the same test samples, and the algorithms whose circle shape is smaller are considered as better.

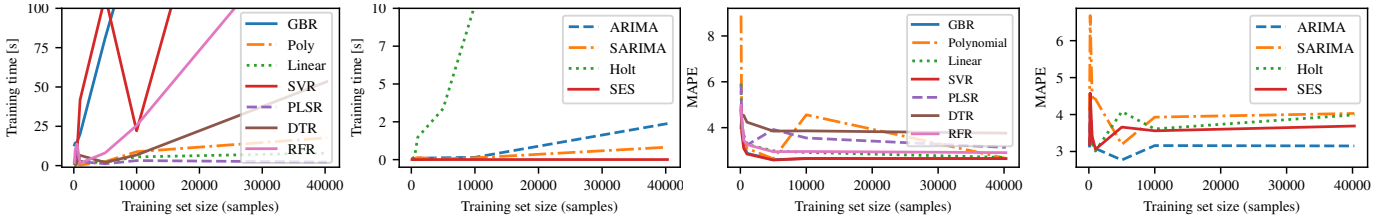


Fig. 5: Training time and error (MAPE) for different training set sizes.

Observation 2. *In some use cases, it is recommended to look for the minimum MAXE, rather than the minimum error, i.e., MAPE.*

Another aspect to be considered is the available time to predict and to train, therefore we study the behavior of the methods for different training set sizes. Figures 5a-b show the training time for ML and TS algorithms respectively. As can be noticed, excepted Holt-Winters, TS algorithms take less time to train data. Furthermore, ML training time is not linear w.r.t. to the size, but it is high for small and big sizes, and low for the medium size.

At the same time, training time must be combined with error in the prediction for a comprehensive analysis of the algorithms. Figures 5c-d shows MAPE for both the TS and ML models. Clearly the more trained data the lower error, however, it is worth noting that for TS methods the error after a minimum around the size of $1,000$, tends to slightly increase. This result suggests using a small training set for this class. On the other hand, for ML algorithms a general decreasing in the error holds.

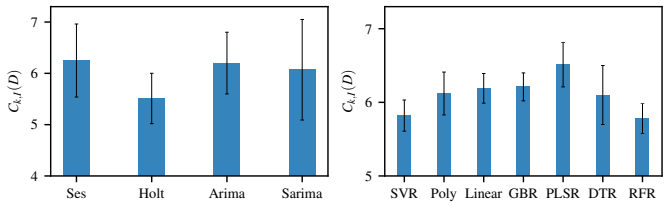
These results confirm our hypothesis of training offline ML algorithms on a large data set, and train the TS methods online Holt-Winters is trained online on a small data-set, with no reduction in the error as proved in Figure 5d.

Observation 3. *Simple methods can be trained online and can address more recent history and more adequate behavior but on a small training set. For more complex (and often more accurate) model an offline training is recommended.*

For this reason, in RoPE the ML models are trained off-line and then used on-line for predicting. The classical model does not need to be trained off-line, and it is better to use more recent data to predict. In this case, there are two major approaches: the sliding window and the expanding window. In the sliding window approach, one uses a fixed size window for training. On the other hand, the expanding window uses more and more training data, while keeping the testing window size fixed. This approach is particularly useful if there is a limited amount of data to work with. Our choice regarding TS is to marry the two methods: start with the expanding window method and, when the window grows sufficiently large, switch to the sliding window method.

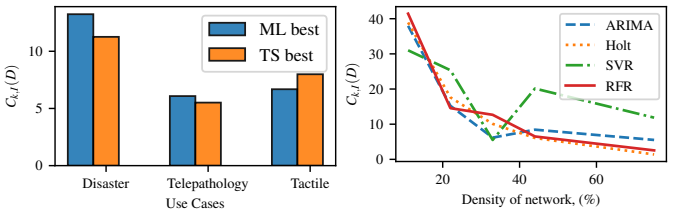
VII. Routing Evaluation

The goal, as mentioned, is to adapt the routing behavior to better cope with the predicted links conditions. Firstly, we need to enumerate the cost function weights used in Eq. 1 to



(a) (b)

Fig. 6: The cost function for the tested algorithms. RFR is the best for ML algorithm, while Holt for TS methods.



(a) (b)

Fig. 7: (a) Comparison of different classes of algorithms for different use cases and (b) algorithms performance among different topologies with increasing connectivity.

take into account specific requirements of different scenarios. Considering in particular our three use cases, we can observe how throughput is really crucial for a Telepathology session, while it is not so relevant for a Disaster response. For Tactile Internet instead, the latency is the predominant factor. For this reason, for the Telepathology application we used ($\alpha = 10^6$, $\mu = 5 \times 10^{-5}$, $\lambda = 10^{-3}$, $\gamma = 2 \times 10^{-9}$), in the Disaster-response use case we used ($\alpha = 10^6$, $\mu = 10^{-6}$, $\lambda = 10^{-5}$, $\gamma = 10^{-12}$), and to emulate Tactile Internet scenarios we used ($\alpha = 10^3$, $\mu = 10^{-4}$, $\lambda = 10^{-3}$, $\gamma = 10^{-10}$).

In the rest of this section, we first evaluate the performance of different prediction algorithms. Then, we compare our approach with existing solutions, and we test the scalability of our strategy and how it can satisfy specific application requirements. Finally, we also run sensitivity experiments by varying some algorithm parameters. The topology we adopted consists of 10 switches and 20 hosts and is inspired by the edge network principles [65].

A. Automate the choice of predictor

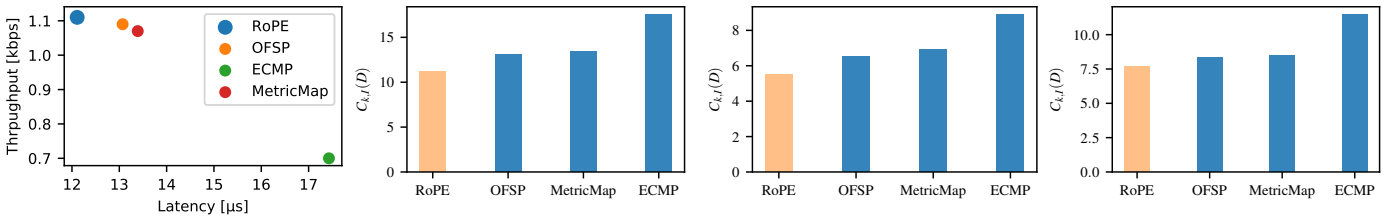
As demonstrated, a prediction method can provide optimal results in a number of cases, but might not work properly in other situations. For this reason, we try to automatically choose the algorithm to apply, in order to guarantee the best possible performance. Choosing the right forecasting method for a given use case is a function of many factors, starting from how much historical data are available, and if exogenous variables (*e.g.*, weather, concerts) play a big role. Moreover, we can consider business needs, whether or not the model needs to be understandable. We imagine this is not always necessary, but we may use a classical method to achieve this requirement.

In the context of our Telepathology use case, the choice of the predictors affects the routing performance (Figure 6). In particular, the TS and ML methods are considered in Figure 6a and Figure 6b, respectively. Our results show that RFR achieves a cost of 5.93, the minimum for the MLs, and Holt-Winters a cost of 5.51, the best for both classes. While our results show that the online training phase has a lower cost than the offline counterpart, this is valid for the considered use case but, in other circumstances, the training offline may result as a valuable strategy.

Figure 7 demonstrates how the approach is general and can handle different use cases and increasing sizes of the network. In particular, Figure 7a shows the cost function value for the three use cases, considering the best TS and ML algorithms for each one. We can see how in a disaster response network, a prediction made by TS algorithms achieves a better transmission quality. This holds because in this scenario fresh data (even if in a small quantity) are more reliable than a huge dataset trained offline, as in ML methods. Conversely, the offline training phase is desirable for tactile Internet applications, where patterns can be discovered in advance and exploited to predict future traffic. This means that, according to user requirements, a class of methods can be preferred to tackle the problem. RoPE is able to detect which class of algorithms to apply and switch among them according to user needs.

In order to generalize our findings, we deployed a more random topology where links among switches and hosts are randomly generated. The number of links between the switches is a parameter in the generation phase and it affects the density of the network. This value is changed to evaluate scalability and test the performance of the framework. Results in the Telepathology case are depicted in Figure 7b, for a different number of links in the network.

On the basis of these findings, the choice of the predictor comprises many factors: use case, expressed as preferences by the user, seasonality of data, frequency in the adaptation of routing, and, consequently, frequency of data collection. Our framework can adequately choose which algorithm to apply, based on the user preferences, for an autonomous network management system. In detail, the choice of the best predictor first selects the class (ML or TS) by evaluating the user needs. TS is used by default for its ability to be trained online and providing an understandable model. Instead, in case the application exhibits patterns that a schema can discover offline, ML is preferred. For example, among the three use cases that we evaluated, ML class provides the best results for Tactile Internet, while TS for Telepathology and Disaster Response. However, other cases can be considered as well. Thanks to the generality of the approach, they can be studied by leveraging the general cost function in order to better identify the proper class. The further comparison is distinct for the two classes as follows. *(i)* For the *TS methods*: on one hand, if marked seasonality is denoted, the system selects ARIMA for the best MAPE (Table II). In fact, ARIMA provides a lower MAPE compared to SARIMA and a comparable training time. For both the algorithms we set the training window to $5,000$ values, since MAPE achieves the minimum at this size for the two methods (Figure 5d). On the other hand, if there is no seasonality, we then investigate the value of r, and if greater than the default value ($20s$), we select Holt-Winters with the training set size of $1,000$ samples as default predictor. In such



(a) (b) (c) (d)

Fig. 8: Comparison for different routing strategies. (a) Trade-off between latency and throughput. Our solution provides an higher throughput and a lower latency. (b)-(c)-(d) The effectiveness of the three routing strategies among the three use cases, Disaster, Telepathology and Tactile respectively.

a way, we select the more accurate method w.r.t SES, but we limit the training set to reduce the training time to a reasonable value (Figure 5b) that can also achieve the best MAPE for this method (Figure 5d). When r is lower than the default, we set SES as the preferred option for its lower training time (Figure 5b) in order to satisfy the more frequent routing updates. *(ii)* For the *ML methods*: our system sets SVR as the predefined predictor method for its lowest MAPE (Figure 5c and Table IV). In this case, the size of training data partially affects the accuracy, and, for this reason, we use as much data as available, since SVR minimizes the MAPE on almost any size of the training set.

Furthermore, the system allows the user to request for any algorithm presented and evaluated in this paper. For example, the user can choose an algorithm with a small MAXE, such as DTR, since a considerable prediction error would lead to erroneous routing selection, with a severe consequence on the application. In this case, a conservative approach may appear to be an effective strategy, and the predictor results to be less precise but almost correct most of the time.

B. Routing Performance Improvement

In this experimental setup we evaluate the quality improvement by comparing our solution against other currently deployed algorithms (Figure 8).

In particular, we compare RoPE against the Equal-Cost-Multi-Path (ECMP), Online Flow Size Prediction (OFSP) [32] and agasint MetricMap [33]. *ECMP*, a well-known algorithm, is used a the baseline. In *OFSP*, authors detect elephant flows by means of the GPR algorithm; hence, the least congested path to route such flows is selected while the ECMP protocol is used to route mice flows. *MetricMap* uses the Very Fast Decision Tree (VFDT) online algorithm [66] to learn and classify traffic. The routing protocol is atop MintRoute and specified for Wireless Networks, but can be generalized.

First, we compare the achieved latency and throughput by using the RFR prediction algorithm for RoPE in Figure 8a. From this result, we can state that RoPE reduces the latency while increasing the application throughput, with respect to the other solutions. The result also points out the flaws of a simple yet deployed approach ECMP, highlighting the benefits brought by an adaptive routing combined with SDN.

Although throughput and latency can be considered as the most major metrics to evaluate, we rely on the cost function (Eq. 1) for a more general evaluation. Figures 8b-c-d depict the function value for the three exposed use cases. As can be seen, RoPE significantly outperforms all the other methods. The resulting routing policy reduced the latency while keeping a stable jitter and high throughput among the three use cases. We can state that our approach yields the best results for the considered applications. We may observe how, while OFSP optimizes the routing for elephant flow that is not long in time, our approach can modify the path even in a second phase, useful for long transmission. Similarly for MetricMap, where online training does not lead to a significantly improved quality.

Observation 4. *Modifying the routing even when the communication is ongoing can improve the application quality.*

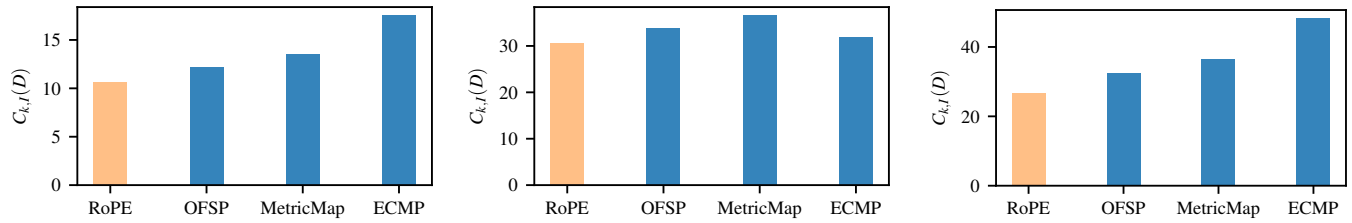
C. Real-case Environment on GENI

To establish the practicality of our approach, we test its scalability over the GENI testbed [67], which provides physical machines and physical links for testing purposes. In particular, we deployed the three applications and the models are retrained over real-world data following the same procedure exposed in Section VI, but the emulated network of Mininet is replaced with physical and virtual links. Based on the previous findings, we select the optimal predictors for each use case and the results are compared against the above state-of-the-art algorithms, as detailed in Figure 9. A comparison between Fig. 8 and 9 shows that conclusions about RoPE in Mininet hold in GENI as well, even though a higher latency and throughput is obtained in real networks. The RoPE cost function is adequately smaller than the state-of-the-art. Moreover, Table V provides details on each component of the cost function for a Disaster Response scenario. As can be seen, no algorithm outperforms the others in all the adopted metrics, but RoPE achieves excellent results in both the latency and the jitter, which leads to an overall better outcome.

TABLE V: Performance evaluation in the context of the Disaster Response application running on the GENI virtual network testbed. Even in these real settings RoPE outperforms related solutions.

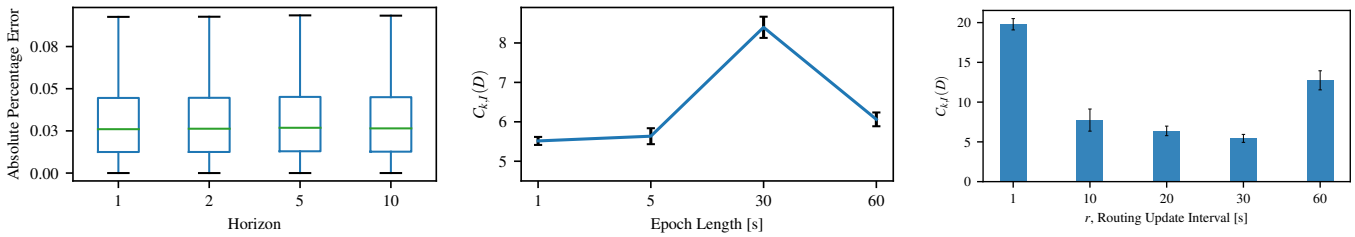
Solutions	Thr. [kbps]	Lat. [μs]	Jit. [ms]	Δ Jit. [ns]	$C_{K,I}(D)$
RoPE	3929.7	10.48	0.83	9.53	**10.58**
OSFP	4107.1	12.07	1.00	7.22	12.16
MetricMap	4077.1	13.39	1.02	9.03	13.48
ECMP	3702.2	17.43	0.96	4.68	17.52

In addition to the evaluation by means of Eq. 1, we also consider the requirements of the applications and we check whether or not these are satisfied by RoPE. In Table VI we compare the specific requirements against results achieved



(a) (b) (c)

Fig. 9: Comparison for different routing strategies over the GENI testbed. Our solution outperforms the related work, as confirmed by (a)-(b)-(c), where the cost function with the proper coefficients is computed for the three use cases, Disaster Scenario, Telepathology and Tactile Internet, respectively.



(a) (b) (c)

Fig. 10: (a) Error vs Horizon for TS methods. (b) Error vs Epoch length for TS and ML algorithm. (c) Routing update interval and cost function.

TABLE VI: Application requirements and the satisfiability achieved by using RoPE.

Use Case	Measurement	Required Value	Obtained Value
Tactile	Latency	1 ms	532.274 μs
Telepathology	Video Bitrate	900 kbps	902.45 kbps
	Latency	100 ms	22.76 ms
Disaster	Jitter	1 ms	0.832 ms
	App. Throughput	3 Mbps	3.929 Mbps

by using RoPE on the GENI testbed. We can notice how RoPE brings benefit even from a user perspective, fulfilling the demands of the applications and enabling the deployment of such services.

Tactile applications entail at least 1-ms latency to work appropriately, hence we select an algorithm that best suits such circumstances, with the help of the cost function. In particular, by using SVR as a predictive algorithm, we can satisfy the requirement and guarantee an adequate service.

Similarly, we select Holt-Winters for the Telepathology use case, where we focus on the achieved bitrate and latency. We deployed an application that sends the video captured by the emulated microscope and sends it to a program responsible for performing video processing [2]. The client sends the video at a maximum bitrate through *ffmpeg* and we measure whether or not the network can provide the adequate throughput. Besides, we desire the latency to be lower than 100 ms to assure the real-time control of the microscope. Holt-Winters was chosen based on previous experiments, and it provides excellent results, as proved in Table VI.

Finally, we select ARIMA for the Disaster Response use case. In order to evaluate the feasibility of applications deployed during a disaster, we implemented a program that continuously sends the recorded audio to a server that processes it and provides useful information such as the presence of humans and the corresponding location. The requirements are selected so that Google libraries used to process audios work best and to enable a fast response. The results (Table VI) reveal that the use of RoPE ensures the application to function properly.

D. Sensitivity Analysis

We also conduct a sensitivity analysis of the performance of predictors with respect to key design parameters for the reference use case of the Telepathology application. (Figure 10).

Horizon. We run tests to study if it is possible to predict more than one future value. The results of figure 10a demonstrates that TS algorithms provide a low error even predicting more than 1-step ahead. In this way, the algorithm can be used to predict more than one single value, ensuring that the bandwidth in the future will be under the threshold for a while. In fact, the graph shows that the prediction error is generally independent of the horizon.

Epoch Length. The Epoch length refers to the time interval in collecting data from the switches. We investigate how the performance of the system changes w.r.t. the bandwidth measurement granularity. Figure 10b shows that a even a not very frequent collection leads to a low cost function, for this reason, we set it to 5-sec.

Routing Update Interval. As discussed in § VII, we want routing to adapt to the network conditions, but without introducing instability over the network. In Figure 10c we evaluate the cost function for different adaptation interval. We can notice that 30 seconds provides the best quality in transmission. This value also guarantees adaptability without incurring in too frequent changes.

VIII. Conclusion

This paper presents RoPE, a new solution to speed up the transfer of critical data. Its main novelty resides in its traffic engineering logic: it predicts the future load on links of a path and then chooses the best one according to data computed. This algorithm allows avoiding congested paths and reduces

delay in the transmission, providing a more effective way of routing critical information with respect to other algorithms existing in the literature. The results confirm the impossibility of one prediction algorithm to fit all the use cases. Apparently, Machine Learning provides excellent results, which reduces the latency in critical communications. However, Time Series can be used for their fast training phase and the straightforward model. In fact, the results suggest that for Disaster response applications TSs are more appropriate.

RoPE leverages SDN features, *e.g.*, centralized controller and the context-based control path, to collect information about the traffic load on the links and takes a new road in case of predicted congestion. Leveraging SDN switches programmability, the framework can quickly react to excessive predicted load on links and adapts the routing to address the congestion. This framework is intended to overcome well-known problems related to edge-based applications, such as latency and throughput requirements. Due to the diversity of applications and data generated, RoPE addresses the user needs by autonomously detecting the data properties, selecting the proper model and applying prediction values to the routing.

Moreover, the paper presents a comprehensive analysis of regression algorithms to evaluate the advantages and disadvantages of the class of methods and depicts the logic behind the presented framework. Possible future work might focus on investigating whether new models can be used in addition to the ones implemented.

ACKNOWLEDGEMENT

This work has been partially supported by NSF under Award Numbers CNS1647084, CNS1836906, and CNS1908574.

REFERENCES

[1] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan 2017.

[2] A. Sacco, F. Esposito, P. Okorie, and G. Marchetto, "Livemicro: An edge computing system for collaborative telepathology," in *Proceedings of the 2nd USENIX Workshop on Hot Topics in Edge Computing*, ser. HotEdge '19. New York, NY, USA: ACM, 2019.

[3] G. Castellano, F. Esposito, and F. Risso, "A distributed orchestration algorithm for edge computing resources with guarantees," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, April 2019, pp. 2548–2556.

[4] D. Chemodanov, P. Calyam, and F. Esposito, "A near optimal reliable composition approach for geo-distributed latency-sensitive service chains," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, April 2019, pp. 1792–1800.

[5] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: ACM, 2016, pp. 272–285.

[6] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: ACM, 2015, pp. 325–338.

[7] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, "Oboe: Auto-tuning video abr algorithms to network conditions," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18. New York, NY, USA: ACM, 2018, pp. 44–58.

[8] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, and K. Winstein, "Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 267–282.

[9] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 197–210.

[10] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016.

[11] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct 2016.

[12] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 2:1–2:23, Feb. 2019.

[13] A. Scalingi, F. Esposito, W. Muhammad, and A. Pescapé, "Scalable provisioning of virtual network functions via supervised learning," in *2019 IEEE Conference on Network Softwarization (NetSoft) (NetSoft 2019)*, Paris, France, Jun. 2019.

[14] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, 2018.

[15] B. Schlinker *et al.*, "Engineering egress with edge fabric: Steering oceans of content to the world," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 418–431.

[16] K.-K. Yap *et al.*, "Taking the edge off with espresso: Scale, reliability and programmability for global internet peering," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 432–445.

[17] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Internet traffic forecasting using neural networks," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, July 2006, pp. 2635–2642.

[18] E. S. Yu and C. Y. R. Chen, "Traffic prediction using neural networks," in *Proceedings of GLOBECOM '93. IEEE Global Telecommunications Conference*, Nov 1993, pp. 991–995 vol.2.

[19] Y. Li *et al.*, "Inter-data-center network traffic prediction with elephant flows," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, pp. 206–213.

[20] Cooperative association for Internet data analysis. [Online]. Available: http://www.caida.org/tools

[21] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 4, pp. 537–549, Aug. 2003.

[22] J. Strauss, D. Katabi, F. Kaashoek, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '03. New York, NY, USA: ACM, 2003, pp. 39–44.

[23] J. Sommers, P. Barford, and W. Willinger, "A proposed framework for calibration of available bandwidth estimation tools," in *11th IEEE Symposium on Computers and Communications (ISCC'06)*, June 2006, pp. 709–718.

[24] T. Benson, A. Anand, A. Akella, and M. Zhang, "Microte: Fine grained traffic engineering for data centers," in *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '11. New York, NY, USA: ACM, 2011, pp. 8:1–8:12.

[25] K. He *et al.*, "Measuring control plane latency in sdn-enabled switches," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, ser. SOSR '15. New York, NY, USA: ACM, 2015, pp. 25:1–25:6.

[26] K. Ilgun, R. A. Kemmerer, and P. A. Porras, "State transition analysis: a rule-based intrusion detection approach," *IEEE Transactions on Software Engineering*, vol. 21, no. 3, pp. 181–199, March 1995.

[27] R. Beverly, K. Sollins, and A. Berger, "Svm learning of ip address structure for latency prediction," in *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, ser. MineNet '06. New York, NY, USA: ACM, 2006, pp. 299–304.

[28] P. Bermolen and D. Rossi, "Support vector regression for link load prediction," *Computer Networks*, vol. 53, no. 2, pp. 191 – 201, 2009.

[29] Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer tcp throughput," *SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 145–156, Aug. 2005.

[30] M. Swany and R. Wolski, "Multivariate resource performance forecasting in the network weather service," in *SC '02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, Nov 2002, pp. 11–11.

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

[32] P. Poupart *et al.*, "Online flow size prediction for improved network routing," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, Nov 2016, pp. 1–6.

[33] Y. Wang, M. Martonosi, and L.-S. Peh, "Predicting link quality using supervised learning in wireless sensor networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 3, pp. 71–83, Jul. 2007.

[34] D. Wolpert, K. Tumer, and J. Frank, "Using collective intelligence to route internet traffic," in *Advances in neural information processing systems*, 1999, pp. 952–960.

[35] S. Kumar and R. Miikkulainen, "Dual reinforcement q-routing: An on-line adaptive routing algorithm," in *Proceedings of the artificial neural networks in engineering Conference*, 1997, pp. 231–238.

[36] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing ospf weights," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 2, March 2000, pp. 519–528 vol.2.

[37] R. Cohen, Y. Dagan, and G. Nakibly, "Proactive rerouting in network overlays," in *2018 IFIP Networking Conference (IFIP Networking) and Workshops*, May 2018, pp. 1–9.

[38] M. Chiesa, G. Rétvári, and M. Schapira, "Lying your way to better traffic engineering," in *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16. New York, NY, USA: ACM, 2016, pp. 391–398.

[39] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia iot systems," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1126–1139, May 2018.

[40] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, March 2014.

[41] C. Kalalas, L. Thrybom, and J. Alonso-Zarate, "Cellular communications for smart grid neighborhood area networks: A survey," *IEEE Access*, vol. 4, pp. 1469–1493, 2016.

[42] M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis, "5g-enabled tactile internet," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 460–473, March 2016.

[43] A. A. Ateya, A. Vybornova, R. Kirichek, and A. Koucheryavy, "Multi-level cloud based tactile internet system," in *2017 19th International Conference on Advanced Communication Technology (ICACT)*, Feb 2017, pp. 105–110.

[44] A. Ateya, A. Muthanna, I. Gudkova, A. Abuarqoub, A. Vybornova, and A. Koucheryavy, "Development of intelligent core network for tactile internet and future smart systems," *Journal of Sensor and Actuator Networks*, vol. 7, no. 1, p. 1, 2018.

[45] R. S. Weinstein, A. R. Graham *et al.*, "Overview of telepathology, virtual microscopy, and whole slide imaging: prospects for the future," *Human Pathology*, vol. 40, no. 8, pp. 1057 – 1069, 2009.

[46] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, Feb 2018.

[47] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, Fourthquarter 2017.

[48] H. Trinh, D. Chemodanov, S. Yao, Q. Lei *et al.*, "Energy-aware mobile edge computing for low-latency visual data processing," in *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, Aug 2017, pp. 128–133.

[49] S. Chung *et al.*, "A novel image-based tool to reunite children with their families after disasters," *Academic Emergency Medicine*, vol. 19, no. 11, pp. 1227–1234, 2012.

[50] J. Wang, J. Pan, and F. Esposito, "Elastic urban video surveillance system using edge computing," in *Proceedings of the Workshop on Smart Internet of Things*, ser. SmartIoT '17. New York, NY, USA: ACM, 2017, pp. 7:1–7:6.

[51] M. Sharifi, S. Kafaie, and O. Kashefi, "A survey and taxonomy of cyber foraging of mobile devices," *IEEE Communications Surveys Tutorials*, vol. 14, no. 4, pp. 1232–1243, Fourth 2012.

[52] J. Burchard, D. Chemodanov, J. Gillis, and P. Calyam, "Wireless mesh networking protocol for sustained throughput in edge computing," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, Jan 2017, pp. 958–962.

[53] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 339–350, Aug. 2013.

[54] M. Alreshoodi and J. Woods, "Survey on QoE\QoS correlation models for multimedia services," *arXiv preprint arXiv:1306.0221*, 2013.

[55] Q. Zhang, J. Liu, and G. Zhao, "Towards 5g enabled tactile robotic telesurgery," *arXiv preprint arXiv:1803.03586*, 2018.

[56] Y.-s. Lim, Y.-C. Chen, E. M. Nahum, D. Towsley, and R. J. Gibbens, "How green is multipath tcp for mobile devices?" in *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges*. New York, NY, USA: ACM, 2014, pp. 3–8.

[57] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[58] P. Geladi and B. R. Kowalski, "Partial least-squares regression: a tutorial," *Analytica Chimica Acta*, vol. 185, pp. 1 – 17, 1986.

[59] M. Xu, P. Watanachaturaporn, P. K. Varshney, and M. K. Arora, "Decision tree regression for soft classification of remote sensing data," *Remote Sensing of Environment*, vol. 97, no. 3, pp. 322–336, 2005.

[60] S. Troia, A. Rodriguez, R. Alvizu, and G. Maier, "Senatus: An experimental sdn/nfv orchestrator," in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2018, pp. 1–5.

[61] FFmpeg documentation. [Online]. Available: https://www.ffmpeg.org/

[62] Unina Traffic and data traces. [Online]. Available: http://www.grid.unina.it/Traces/index.php

[63] F. Pedregosa, G. Varoquaux, A. Gramfort *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, pp. 2825–2830, Nov. 2011.

[64] J. H. F. Flores, P. M. Engel, and R. C. Pinto, "Autocorrelation and partial autocorrelation functions to improve neural networks models on univariate time series forecasting," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, June 2012, pp. 1–8.

[65] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, thirdquarter 2017.

[66] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '01. New York, NY, USA: ACM, 2001, pp. 97–106.

[67] Geni, Exploring Networks of the Future. [Online]. Available: https://www.geni.net/

Alessio Sacco received the M.Sc. degree in Computer Engineering from the Politecnico di Torino, where he is currently pursuing the Ph.D. degree in Computer Engineering. His research interests include architecture and protocols for network management; implementation and design of cloud computing applications; algorithms and protocols for service-based architecture, such as Software Defined Networks (SDN), used in conjunction with Machine Learning algorithms.

Flavio Esposito is an Assistant Professor with the Department of Computer Science at Saint Louis University (SLU). He also has an affiliation with the Parks College of Engineering at SLU. He received an M.Sc. degree in Telecommunication Engineering from the University of Florence, Italy, and a Ph.D. in computer science from Boston University in 2013. Flavio worked in the industry for a few years, and his main research interests include network management, network virtualization, and distributed systems. Flavio is the recipient of several awards, including four National Science Foundation awards and two best paper awards, one at IEEE NetSoft 2017 and one at IEEE NFV-SDN 2019.

Guido Marchetto received the Ph.D. degree in computer engineering from the Politecnico di Torino, in 2008, where he is currently an Associate Professor with the Department of Control and Computer Engineering. His research topics cover distributed systems and formal verification of systems and protocols. His interests also include network protocols and network architectures.