Virtual Network Topology Adaptability Based on Data Analytics for Traffic Prediction

Fernando Morales, Marc Ruiz, Lluís Gifre, Luis M. Contreras, Víctor López, and Luis Velasco

***Abstract*—The introduction of new services requiring large and dynamic bitrate connectivity can cause changes in the direction of the traffic in metro and even core network segments throughout the day. This leads to large overprovisioning in statically managed virtual network topologies (VNTs), which are designed to cope with the traffic forecast. To reduce expenses while ensuring the required grade of service, in this paper we propose a VNT reconfiguration approach based on data analytics for traffic prediction (VENTURE). It regularly reconfigures the VNT based on the predicted traffic, thus adapting the topology to both the current and the predicted traffic volume and direction. A machine learning algorithm based on an artificial neural network is used to provide robust and adaptive traffic models. The reconfiguration problem that takes as its input the traffic prediction is modeled mathematically, and a heuristic is proposed to solve it in practical times. To support VENTURE, we propose an architecture that allows collecting and storing data from monitoring at the routers and that is used to train predictive models for every origin-destination pair. Exhaustive simulation results of the algorithm, together with the experimental assessment of the proposed architecture, are finally presented.**

***Index Terms*—Machine learning for traffic prediction; Multilayer networks; Virtual network topology reconfiguration.**

I. Introduction

Static virtual network topologies (VNTs), where large packet-switching nodes (e.g., IP/MPLS routers) are connected through virtual links (*vlinks*) supported by static connections in the optical layer, have been commonly designed to cope with the off-net traffic forecast. However, the introduction of cloud infrastructure in the telecom operators' networks to create the telecom cloud [1] facilitates the introduction of new types of services (e.g., live TV and video distribution [2]) that require large bitrate connectivity and cause changes in the direction of the traffic throughout the day in metro and even core network segments. This, together with the overall traffic increment operators' networks are needing to deal with year after year, means static packet network topologies are largely overprovisioned, thus increasing the network total cost of ownership (TCO). In view of that, network operators are looking for more efficient architectures able to reduce TCO while providing the required grade of service. To that end, VNTs need to be dynamically adapted not only to variations in volume, but also to changes in the direction of the traffic.

To support connectivity dynamicity, the IETF has recently standardized the application-based network operation (ABNO) architecture [3], which includes, among others, (i) the ABNO controller as the entrance point to the network for provisioning and advanced network coordination, which acts as a system orchestrator, invoking its inner components according to a specific workflow; (ii) a virtual network topology manager (VNTM) in charge of reconfiguring on-demand VNTs; (iii) a path computation element (PCE) to compute the path for new label-switched paths (LSPs) on the traffic engineering database (TED); (iv) a provisioning manager (e.g., an SDN controller) responsible for managing LSPs, both at the optical layer (lambda-switch capable, LSC) and at the packet layer (packet-switch capable, PSC); and (v) the operations, administration, and maintenance (OAM) handler to receive notifications and monitored counters.

To automate VNT adaptability, traffic needs to be monitored in the packet nodes and counters need to be accessible by the OAM handler. In particular, the disaggregated traffic volume forwarded by each packet node to every other destination node should be available. In addition, notifications can also be triggered when the used vlink capacity reaches some configured threshold (e.g., 90%). In fact, threshold-triggered VNT reconfiguration where the OAM handler receives notifications from the control plane and reroutes individual IP/MPLS paths (PSC LSPs) in a reactive manner was proposed in [4].

However, the number of transponders to be installed tends to be as high as in a static VNT since dimensioning must be done to cope with the maximum daily traffic forecast during a planning period (e.g., one year). In view of that, we propose an alternative approach to adapt the

Manuscript received June 14, 2016; revised September 1, 2016; accepted September 1, 2016; published October 7, 2016 (Doc. ID 268095).

F. Morales (e-mail: fmorales@ac.upc.edu), M. Ruiz, L. Gifre, and L. Velasco are with the Optical Communications Group (GCO) at Universitat Politècnica de Catalunya (UPC), Barcelona, Spain.

L. M. Contreras and V. López are with Telefonica R&D (TID), Distrito Telefonica, Madrid, Spain.

http://dx.doi.org/10.1364/JOCN.9.000A35

VNT based on a traffic prediction that can reduce the number of transponders to be deployed in scenarios where traffic variations are as a result of changes in traffic directionality.

Because of its benefits, VNT reconfiguration has been widely studied in the literature. The authors in [5] proposed a centralized path reallocation module running periodically and aimed at minimizing the number of used transponders. To follow traffic changes, the authors of [6] proposed to add/remove one single lightpath each time the VNT is reconfigured. Another topic is using monitored data to produce estimations that can help to anticipate changes in the traffic and proactively reconfigure the VNT beforehand. In that regard, the authors of [7] proposed a method for reducing errors in traffic estimations, while the authors of [8] used estimated traffic to predict predefined scenarios. Finally, the authors of [9] used future traffic estimations to trigger a VNT reconfiguration after the detection of an anomalous amount of traffic between two nodes.

VNT reconfiguration requires powerful algorithms to analyze large amounts of traffic monitoring data to anticipate, when possible, traffic changes. In this paper, we propose using big data analytics to periodically (e.g., every hour) predict traffic. In the case that the VNT needs to be reconfigured, the predicted traffic is used as the input of a VNT optimizer that finds the topology for the next period, thus implementing a decision-making process based on the *observe-analyze-act* loop [10]. Specifically, the contribution of this paper is two-fold:

1) Targeting the adaptation of the VNT to future traffic conditions, in Section III we devise a robust and adaptive artificial neural network (ANN) model that is afterward used as the input of the VNT reconfiguration problem. We call this approach the *VNT reconfiguration based on traffic prediction* (VENTURE). The VENTURE problem is formally stated and formulated as an ILP model in Section IV; in view of its complexity, a heuristic algorithm providing better trade-off between complexity and optimality is finally designed.
2) A big data network manager architecture to support VNT adaptability based on traffic predictions from applying data analytics to the monitored traffic data, as well as a workflow assuming the ABNO architecture proposed in Section V.

The discussion is supported by results from an exhaustive simulation over a realistic scenario and from the experimental validation of the big data-based architecture in Section VI.

II. VNT Design and Reconfiguration Options

As introduced above, several approaches to design and dynamically reconfigure the VNT can be devised. In this section, let us first review two of them: (i) the *static VNT design* and (ii) the *threshold-based VNT capacity reconfiguration*.

In the static VNT design, the topology is designed and dimensioned to cope with the maximum daily traffic forecast for every origin-destination (OD) pair during a planning period. The resulting topology is thus capable of supporting the traffic at any time during that period provided it has a perfect traffic forecast. Figure 1 presents an example of a seven-node VNT, where the capacity of every vlink supports the maximum daily traffic volume. For illustrative purposes, the plot in Fig. 1(a) shows the variability in link 6-1 that needs to be dimensioned with a capacity of 200 Gb/s. Figure 1(b) shows the resulting VNT with the capacity of every vlink. It is clear, in view of Fig. 1(a), that the main drawback of the static VNT design is overprovisioning, since most of the available capacity in the VNT will remain underutilized throughout the day.

To reduce capacity overprovisioning, the vlinks can be adapted over time instead of allocating a constant amount of resources. Let us assume that the capacity of the existing vlinks can be increased and decreased to follow the traffic variations but no new vlinks can be created or removed, keeping the VNT invariant. Traffic can be monitored at IP routers, and when the amount of traffic through a vlink reaches some threshold (e.g., 90%), the network controller can increase the capacity of such a vlink by setting up a parallel lightpath between the two IP routers; conversely, the unused capacity can be released by tearing down lightpaths.

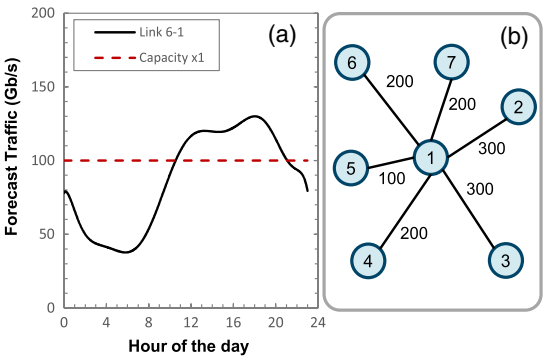

Fig. 1. Static VNT design.

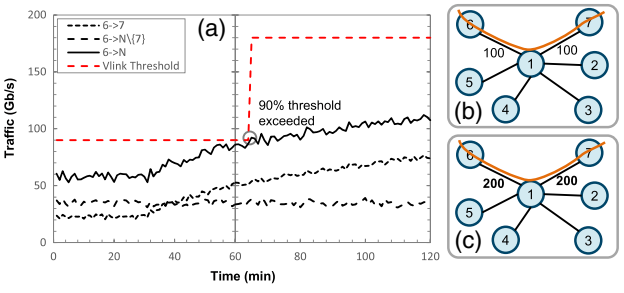

Fig. 2. Threshold-based VNT capacity reconfiguration.

The example in Fig. 2(a) presents monitored traffic data captured during the last 2 h in node 6, where traffic from that node to every other node in the VNT (labeled as $6 \rightarrow N$), from node 6 to node 7 ($6 \rightarrow 7$), and from node 6 to every other node except to node 7 ($6 \rightarrow N \backslash \{7\}$) is plotted. Figure 2(b) shows the initial VNT where every vlink is supported by a 100 Gb/s lightpath in the underlying optical layer; the IP/MPLS path for OD 6-7 is also shown. A 90% threshold is configured and, in the event of threshold violation, the capacity of some vlinks is increased. In our example, two threshold violations for vlinks 1-6 and 1-7 are received, so the VNT capacity is updated [Fig. 2(c)]. It is worth noting that the IP/MPLS path for OD $6 \rightarrow 7$ is not affected by the VNT reconfiguration. As shown in the example, the threshold-based reconfiguration is able to adapt the VNT capacity to traffic changes, so resources in the optical layer are allocated only when the vlinks need to increase their capacity. However, the same number of transponders as in the static VNT design approach need to be installed in the IP routers; for instance, in the example in Fig. 2, two transponders are installed in routers 6 and 7, and another four in router 1 are reserved for vlinks 1-6 and 1-7.

Let us assume now that, instead of monitoring vlink capacity usage, the OD traffic is monitored in the routers. Indeed, by analyzing the plots in Fig. 2(a), we realize that traffic $6 \rightarrow 7$ is responsible for the registered traffic increment. In this case, let us assume that new vlinks can be created/removed in addition to increasing the capacity of the existing links, so the VNT is actually changed. We propose an approach where the OD traffic is periodically analyzed and the current VNT is reconfigured accordingly. An example following this approach is illustrated in Fig. 3, where the OD traffic $6 \rightarrow 7$ is analyzed at $t = 60$ and a maximum value (e.g., 90 Gb/s) is predicted for the next hour. Then, a new vlink between nodes 6 and 7 can be created by establishing a lightpath on the optical layer and having traffic $6 \rightarrow 7$ rerouted [Fig. 3(b)]. Note that this solution reduces two transponders to be installed in router 1 compared to the previous approaches. It is clear that this reduction will happen when the amount of traffic is large enough. In particular, when the amount of traffic exceeds the capacity of the installed transponders (e.g., 100 Gb/s) direct vlinks can be created for part of that traffic, while the residual part could be routed through a different IP/MPLS path.

In order to adapt the VNT to changes in the traffic, we propose a predictive model built upon the monitored OD traffic data. For every OD pair, meaningful statistical values are predicted (e.g., the maximum bitrate for the next hour) and used to adapt the VNT to meet the future traffic matrix, assuming that every OD traffic can be conveyed through two IP/MPLS paths at the most. We call this approach VENTURE.

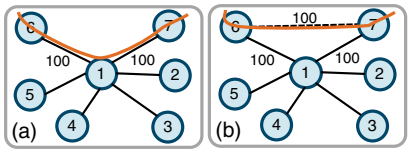

Fig. 3. VNT reconfiguration.

III. Data Analytics-Based VNT Adaptation

In this section, we present the proposed modules and the machine learning procedure for the VENTURE approach. We assume that traffic monitoring data are collected at the edge IP routers at regular intervals, e.g., every 15 min. Every edge router collects a set of samples for the traffic to every other destination router, which is stored in a *collected data repository* (Fig. 4). Note that since we focus on OD traffic monitoring, $|N| \times (|N| - 1)$ traffic samples need to be stored at every monitoring interval, where $|N|$ is the number of routers.

Following a predefined time period, e.g., every hour, a time series from the collected data repository is retrieved for each OD pair and pre-processed applying data stream mining *sketches* to conveniently summarize collected data, thus producing modeled data representing the OD pair that is stored in a *modeled data repository*. Modeled data includes, among others, for every OD the minimum, maximum, average, and last collected bitrate within the hour.

The set of modeled variables for the current period t is stored in a repository, together with variables belonging to previous periods. A *prediction module* based on machine learning techniques generates the OD traffic matrix predicted for the next period that is used by a *decision-maker* module to decide whether the current VNT needs to be reconfigured. In case a reconfiguration needs to be performed, the current and the predicted OD traffic matrices are provided to the *VNT optimizer* to adapt the VNT. Once the algorithm finds a solution, the network controller would be responsible for implementing the changes in the data plane.

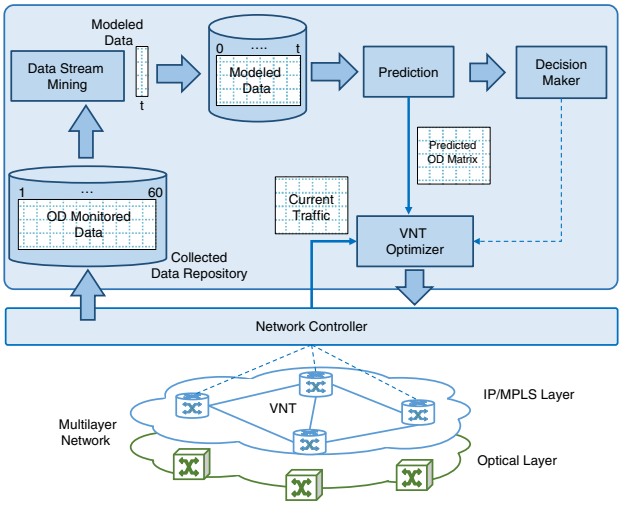

Fig. 4. Applying the *observe-analyze-act* loop for VNT reconfiguration.

The prediction module consists of ANN-based models [11], selected because of their inherent capability to adapt to traffic changes in a non-supervised manner; we consider different ANNs to separately predict the traffic of each OD pair. Each ANN receives as input p previous maximum bitrate measurements of the corresponding OD pair from the modeled data repository and returns its expected maximum bitrate at time t. Note that considering maximum instead of the average bitrate allows us to adapt the VNT to the maximum expected traffic, hence ensuring a better grade of service.

Since the size of an ANN depends on the number of inputs, hidden layers, and neurons, we consider ANN models with p inputs, s neurons in a single hidden layer, and one output. Consequently, $s \times (p+1)$ coefficients need to be found to specify each ANN. Aiming at keeping the number of coefficients small, we designed the algorithm shown in Fig. 5 so that it has to be triggered every time an ANN needs to be refitted. It consists of three phases: (i) input data pre-processing, (ii) selection of significant inputs, and (iii) dimensioning of the hidden layer.

In the first phase, a time series X with the maximum bitrate for the selected OD pair is retrieved from the modeled data repository. The *auto-correlation function* (ACF) is applied to X and a list of *lags* is returned, where the ith lag contains the average correlation between each value in the time series and its ith previous value. Based on the lag analysis, a method is triggered to detect whether a periodic repetitive (seasonal) pattern is observable in X [12]. The resulting period *per* defines the number of inputs of the ANN; in the case of non-seasonal data without observable periodical behaviors, we assume $per = 24$ (i.e., one day) for convenience. Once *per* is obtained, X is transformed into a dataset D used for ANN fitting. Every row in D corresponds to a time t within the time series, and every column corresponds to a lag within *per*.

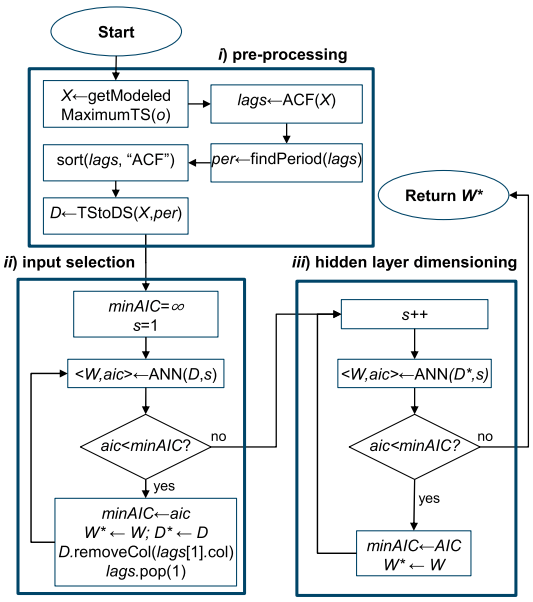

Fig. 5. Self-learning ANN fitting algorithm.

The second phase is an iterative procedure that finds the ANN with the best trade-off between accuracy and number of inputs. This trade-off is captured numerically by the *Akaike information criterion* (AIC) [11]. Starting with $p = per$, the ANN routine fits an ANN from dataset D and returns the corresponding AIC value. While the AIC value obtained improves the lowest one obtained so far, the best ANN is stored and p is decremented, effectively removing one input. Aiming at reducing the complexity of selecting the input to be removed, we select the lag with the lowest ACF. When the minimum AIC is reached, the third phase is executed to further increase the accuracy of the model by adding hidden neurons until the AIC does not improve. The best ANN is eventually returned.

We turn next to the VNT reconfiguration (VENTURE) problem to be solved in the VNTM.

IV. VENTURE Problem

In this section, we first formally state the VENTURE problem and devise an ILP to model it. In light of the complexity of the problem, a heuristic algorithm is eventually devised.

A. Problem Statement

Given:

- The current VNT represented by a graph $G(N, E')$, with N being the set of routers and $E' \subseteq E$ the set of current vlinks. Set E is the set of all possible vlinks connecting two routers.
- The set P with the transponders available in the routers; every transponder has a capacity B.
- The current traffic matrix D.
- The predicted traffic matrix OD. The bitrate b_o of OD pair o must be served following one single path. Only in the case that b_o is enough to fill transponders with an amount over a given boundary usage *tbu* can the bitrate of pair o be split into two flows and served through different paths.

Output: The reconfigured VNT $G^*(N, E^*)$, where $E^* \subseteq E$, and the paths for the traffic on G^*.

Objective: Maximize current and predicted served traffic matrices while minimizing the total number of transponders used.

B. Mathematical Formulation

Note from the problem statement that both the current and the predicted traffic matrices must be served. Consequently, we generate an input traffic matrix OD, where every pair o is the maximum of both the current

and the predicted traffic. In addition, a parameter k_o will be used to specify whether pair o can be served using two paths.

The following sets and parameters are defined:

Topology:

N	set of routers, index n.
E	set of all possible vlinks, index e.
$E^+(n)$	subset of E with vlinks outgoing from router n.
$E^-(n)$	subset of E with vlinks incident in router n.

Traffic:

OD	set of origin-destination pairs, index o. Every o is defined by the tuple $\langle s_o, t_o, d_o, b_o \rangle$, where s_o and t_o specify the source and target nodes, d_o the currently served bitrate and b_o the maximum of current and predicted bitrate to serve for pair o, respectively.
k_o	maximum number of paths to serve pair o; $k_o = 2$ if $b_o \geq tbu$; $k_o = 1$ otherwise.

Equipment:

P	set of transponders, index p. Every transponder consists of one transmitter (tx) and one receiver (rx).
$P^+(n)$	subset of tx transponders in router n.
$P^-(n)$	subset of rx transponders in router n.
$P(n)$	subset of transponders in n. $P(n) = P^+(n) U P^-(n)$.
B	capacity of every transponder.

The decision variables are as follows:

x_p	binary, 1 if transponder p is used, 0 otherwise.
x_{pe}	binary, 1 if transponder p is used to support vlink e, 0 otherwise.
x_{ok}	integer, fraction of bitrate of pair o served through path k.
x_{oke}	integer, fraction of bitrate of pair o served through path k using vlink e.
z_{oke}	binary, 1 if pair o is routed using path k through vlink e, 0 otherwise.
y_n	integer+, number of transponders used at router n.
v_o	integer+, fraction of unserved bitrate of pair o.

Then, the proposed ILP formulation is as follows:

$$\min(|P|+1) \cdot \sum_{o \in OD} v_o + \sum_{n \in N} y_n, \tag{1}$$

subject to:

$$\sum_{e \in E^+(n)} z_{oke} - \sum_{e \in E^-(n)} z_{oke} = \begin{cases} 1 & \forall o \in OD, \quad k = 1, \ldots, k_o, \quad n = s_o \\ 0 & \forall o \in OD, \quad k = 1, \ldots, k_o, \\ & n \in N \backslash \{s_o, t_o\} \\ -1 & \forall o \in OD, \quad k = 1, \ldots, k_o, \quad n = t_o \end{cases}, \tag{2}$$

$$x_{oke} \leq b_o \cdot z_{oke} \quad \forall o \in OD, \qquad k = 1, \ldots, k_o, \qquad e \in E, \tag{3}$$

$$x_{oke} \leq x_{ok} \qquad \forall o \in OD, \qquad k = 1, \ldots, k_o, \quad e \in E, \tag{4}$$

$$x_{ok} - b_o \cdot (1 - z_{oke}) \leq x_{oke} \quad \forall o \in OD, \quad k = 1, \ldots, k_o, \quad e \in E, \tag{5}$$

$$\sum_{k=1}^{k_o} x_{ok} + v_o \geq b_o \quad \forall o \in OD, \tag{6}$$

$$\sum_{k=1}^{k_o} x_{ok} \geq d_o \quad \forall o \in OD, \tag{7}$$

$$\sum_{o \in OD} \sum_{k=1}^{k_o} x_{oke} \leq B \cdot \sum_{p \in P^+(i)} x_{pe} \quad \forall e = (i,j) \in E | i,j \in N, \tag{8}$$

$$\sum_{o \in OD} \sum_{k=1}^{k_o} x_{oke} \leq B \cdot \sum_{p \in P^-(j)} x_{pe} \quad \forall e = (i,j) \in E | i,j \in N, \tag{9}$$

$$\sum_{e \in E^+(n)} x_{pe} \leq x_p \quad \forall n \in N, \qquad p \in P^+(n), \tag{10}$$

$$\sum_{e \in E^-(n)} x_{pe} \leq x_p \quad \forall n \in N, \qquad p \in P^-(n), \tag{11}$$

$$\sum_{p \in P^+(n)} x_p \leq y_n \quad \forall n \in N, \tag{12}$$

$$\sum_{p \in P^-(n)} x_p \leq y_n \quad \forall n \in N. \tag{13}$$

The multi-objective cost function [Eq. (1)] minimizes both the unserved traffic and used transponders, where the highest cost corresponds to the first term.

The network flow constraints in Eq. (2) define paths on the topology for every OD pair. Each of these paths has a continuous capacity assignment along its route, as imposed by Eqs. (3)–(5). Equation (6) allows serving only a fraction of the total capacity b_o of every OD pair; that fraction has to include at least the currently served bitrate, as stated in Eq. (7). Note that optimal solutions might include loops that can be safely removed in a post-processing phase.

Equations (8)–(13) deal with the transponder equipment. Equations (8) and (9) assign transmission and reception transponders to vlinks, respectively, to support the capacitated paths. Equations (10) and (11) prevent someone from assigning one transponder to multiple vlinks. Finally, Eqs. (12) and (13) compute the maximum between the number of transponders used for transmission and reception at every node, which represents the number of transponders to be installed in every router.

ILP problems belong to the NP-complete complexity class, as proven in [13]. The size of the proposed formulation is $O(|N|^4 + |P| \times |N|^2)$ variables and $O(|N|^4 + |P| \times |N|)$ constraints. As an example, the size of the above formulation for the network instance with $k_o = 2$ and 14 nodes presented in Section VI is of 2×10^5 variables and

10^5 constraints. As a result, solving the proposed formulation becomes impractical for realistic scenarios even using commercial solvers; in our tests, solving times were longer than 10 h. Consequently, we developed a heuristic algorithm that provides a much better trade-off between optimality and complexity.

C. Heuristic Algorithm

We devise a heuristic algorithm to solve the VENTURE problem consisting of three phases; the pseudocode is presented in Table I. After deallocating current traffic and releasing used resources (lines 2–5 in Table I), bitrate b_o of OD pairs is split into two different flows and stored in set Q: flow g_o carries enough bitrate to fill transponders with an amount over tbu, and flow l_o carries the remaining bitrate; these flows will be routed through different paths (lines 6–7). Next, the first two phases focus on routing every flow g_o through the direct vlink connecting source and destination routers (lines 8–9); set F stores the path of the flows. The first phase selects those flows for which a direct vlink already exists in the current VNT, and the second phase does the same for the rest of the flows, thus creating new direct vlinks. After these two phases, the residual bitrate u_o is checked and stored in set U. If all traffic has already been served, the algorithm ends (lines 10–12); otherwise, OD pairs are sorted by the amount of unserved bitrate and the third phase eventually routes the unserved bitrate by possibly increasing the capacity of existing vlinks or by adding new ones (line 13). The reconfigured VNT and the new routing are eventually returned.

The algorithm for the first phase is detailed in Table II. The uncapacitated original topology is used to route flows g_o through an existing direct vlink so as to introduce inertia to the changes in the current topology (line 3 in Table II). The number of transponders to be allocated in the end routers of the direct vlink is computed as the minimum between the amounts of transponders needed to allocate g_o and the unused transponders (lines 4–7); those transponders are allocated to add capacity to the direct vlink (line 8) and a shortest path is computed on the resulting VNT (line 9). In the case where a path is found (i.e., capacity was added to the direct vlink), the path is allocated and the amount of served bitrate is reduced from the one requested (lines 10–13). The updated set Q and the found paths stored in set F are eventually returned (line 14).

The second phase is similar to the first phase, but for flows g_o through a non-existing direct vlink. New capacitated direct vlinks are thus added to the topology to support those flows.

In the third phase, the current topology is extended to a full mesh topology by adding uncapacitated vlinks (lines 2–4 in Table III). Next, a randomized routing procedure is run for a given number of iterations (lines 6–31); at every iteration, the initial extended topology and the unserved bitrate are cloned and the latter randomly sorted, giving higher priority to flows with higher remaining traffic (lines 7–12). Those flows with unserved bitrate are routed using one single path (lines 13–16). Aiming at minimizing the number of used transponders, link metrics are set proportional to the number of transponders needed to allocate the remaining capacity of the current flow (line 15). If no path is found, the corresponding cost is added to the iteration cost (lines 17–19); otherwise, in case the path capacity does not serve the remaining bitrate, we check whether the capacity of its links can be increased using the available resources, i.e., transponders in the end nodes and spectral resources to create a lightpath at the optical layer (lines 20–22). Finally, the path is allocated and the remaining bitrate of the flow is updated, as well as the iteration cost (lines 23–26).

Once a solution has been built, a local search procedure is executed (line 27) aiming at finding a local minimum. The best topology and the found paths are returned as final solutions (lines 29–32).

The local search procedure tries to reduce the total number of used transponders during the constructive phase. Since the number of transponders used in a node is

TABLE I
MAIN ALGORITHM

INPUT $G(N,E'),D,OD,B,tbu$
OUTPUT G^*, F

1: $Q \leftarrow \emptyset$, $U \leftarrow \emptyset$
2: **for each** $d \in D$ **do** dealloc (G,d)
3: **for each** $e \in E'$ **do**
4: setCapacity $(e,0)$
5: releaseTransponders (e)
6: **for each** $o \in OD$ **do**
7: $Q \leftarrow Q \bigcup \{\langle o, g_o, l_o\rangle = \text{split}\, OD(o,B,tbu)\}$
8: $\langle Q, F'\rangle \leftarrow$ PhaseI(G,Q)
9: $\langle G', Q, F''\rangle \leftarrow$ PhaseII(G,Q)
10: **for each** $q \in Q$ **do**
11: $U \leftarrow U \bigcup \{\langle o, u_o = g_o + l_o\rangle\}$
12: **if** $U = \emptyset$ **then return** $\langle G', F' \bigcup F''\rangle$
13: $\langle G^*, F'''\rangle \leftarrow$ PhaseIII(G', U, thr)
14: **if** $F''' = \emptyset$ **then return** INFEASIBLE
15: **return** $< G^*, \langle G^*, F' \bigcup F'' \bigcup F'''\rangle$

TABLE II
PHASE I ALGORITHM

INPUT $G(N,E),Q$
OUTPUT Q, F

1: $F \leftarrow \emptyset$
2: **for each** $q = \langle o, g_o, l_o\rangle \in Q$ **do**
3: **if** $e = (s_o, t_o) \notin E$ OR $g_o = 0$ **then continue**
4: $n_o \leftarrow \text{ceil}(g_o/B)$
5: $n_s \leftarrow$ getNumUnusedTransponders (s_o, P^+)
6: $n_t \leftarrow$ getNumUnusedTransponders (t_o, P^-)
7: $n \leftarrow \min\{n_o, n_s, n_t\}$
8: allocateTransponders (e,n)
9: $f \leftarrow \text{SP}(G, o, g_o)$
10: **if** $f \neq \emptyset$ **then**
11: allocate (G,f)
12: $F \leftarrow F \bigcup \{f\}$
13: $g_o \leftarrow g_o - f.b$
14: **return** $\langle Q, F\rangle$

TABLE III
PHASE III ALGORITHM

INPUT $G(N,E), U, thr$
OUTPUT $G^*(N,E^*), F$

1: $G^*(N,E^*) \leftarrow G(N,E); F \leftarrow \emptyset$
2: **for each** $e=(i,j) \notin E | i,j \in N, i \neq j$ **do**
3: $E^* \leftarrow E^* \bigcup \{e\}$
4: setCapacity$(e,0)$
5: $bestCost \leftarrow +\infty$
6: **for** $ite = 1, \ldots, maxIter$ **do**
7: $iteUnserved \leftarrow 0$
8: $G_{ite} \leftarrow G$
9: $U_{ite} \leftarrow U$
10: $F_{ite} \leftarrow \emptyset$
11: **for each** $u \in U_{ite}$ **do** u.order $\leftarrow$ rand$(0,1) * u_o$
12: sort $(U_{ite}, u.\text{order}, \text{DESC})$
13: **for each** $u \in U_{ite}$ **do**
14: **if** $u.u_o = 0$ **then continue**
15: updateMetrics $(G_{ite}, u.u_o)$
16: $f \leftarrow \text{SP}(G_{ite}, u.o, u.u_o)$
17: **if** $f = \emptyset$ **then**
18: $iteCost \leftarrow iteCost + u.u_o^*(|P|+1)$
19: **continue**
20: **if** $f.b < u.u_o$ AND canIncreaseCap $(f, G_{ite}, u.u_o)$ **then**
21: increaseCap $(f, E_{ite}, u.u_o)$
22: $f.b \leftarrow u.u_o$
23: $F_{ite} \leftarrow F_{ite} \bigcup \{f\}$
24: allocate (G_{ite}, f)
25: $u.u_o \leftarrow u.u_o - f.b$
26: $iteCost \leftarrow iteCost + u.u_o^*(|P|+1)$
27: $\langle G_{ite}, F_{ite} \rangle \leftarrow$ doLocalSearch(G_{ite}, F_{ite})
28: $iteCost \leftarrow iteCost +$ numUsedTransponders(G_{ite})
29: **if** $iteCost < bestCost$ **then**
30: $bestCost \leftarrow iteCost$
31: $< G^*, \langle G^*, F \rangle \leftarrow \langle G_{\text{ite}}, F_{\text{ite}} \rangle$
32: **return** $\langle G^*, F \rangle$

TABLE IV
LOCAL SEARCH PROCEDURE

INPUT $G(N,E), F$
OUTPUT $G^*(N,E^*), F^*$

1: $G^* \leftarrow G; F^* \leftarrow F; E_{rem} \leftarrow E$
2: **while** $E_{rem} \neq \emptyset$ **do**
3: **for each** $e \in E_{rem}$ **do**
4: $e.balance \leftarrow computeTransponderBalance(e)$
5: sort $(E_{rem}, e.balance, \text{DESC})$
6: $e \leftarrow$ removeFirst(E_{rem})
7: $F_e \leftarrow$ getPaths(e)
8: $\langle G_{aux}, F_{aux} \rangle \leftarrow \langle G^*, F^* \rangle$
9: release(G_{aux}, F_e)
10: $E_{aux} \leftarrow E_{aux}\}\{\text{e}\}$
11: sort$(F_e, f.b, \text{DESC})$
12: $allRerouted \leftarrow$ **true**
13: **for each** $f \in F_e$ **do**
14: recomputeZeroCostLinks (G_{aux})
15: $f' \leftarrow \text{SP}(G_{aux}, o, f.b)$
16: **if** $f' = \emptyset$ **then**
17: $allRerouted \leftarrow$ false; **break**
18: allocate(G_{aux}, f')
19: $F_{aux} \leftarrow (F_{aux} \backslash \{f\}) \bigcup \{f'\}$
20: **if** $allRerouted$ **then** $\langle G^*, F^* \rangle \leftarrow \langle G_{aux}, F_{aux} \rangle$
21: **return** $\langle G^*, F^* \rangle$

TABLE V
TIME COMPLEXITY OF THE ALGORITHMS

Phase I/II	Phase III (per iter)	Local Search
$O(\|N\|2 \cdot R0)$	$O(\|N\|2 \cdot (\|E\| \cdot \log\|N\| + R0))$	$O(\|E\|2 \cdot (\log\|E\| + \|Fe\| \cdot \log\|N\|))$
<1 m	298 ms	234 ms

computed as the maximum between transmission and reception, this procedure focuses on releasing transponders that actively contribute to that maximum at every node. The algorithm is detailed in Table IV, where all current vlinks in the VNT are processed (lines 2–20). The vlink with ports most actively contributing to the cost function is selected along with the set of paths routed through it (lines 6–7). This set is released from the VNT and sorted with respect to the bitrate (lines 8–11). Next, the set of paths is re-routed by possibly using new vlinks at zero objective cost (lines 13–15). In the case where a feasible solution is found, the VNT is updated with these changes (line 20).

Table V presents the time complexity of the proposed algorithms, where $R0$ is the worst-case time complexity to find a feasible lightpath to support every direct vlink. For illustrative purposes, the computation time for the scenario with 14 nodes presented in Section VI is also provided.

V. PROPOSED ARCHITECTURE AND WORKFLOW

To support the generic modules introduced in Section III for VENTURE, we propose the architecture in Fig. 6. A big data repository consisting of a distributed database and a data collector is used to store collected data. ABNO's OAM handler includes data stream mining sketches, the modeled data repository, the prediction module running the proposed ANN to anticipate next period traffic conditions,

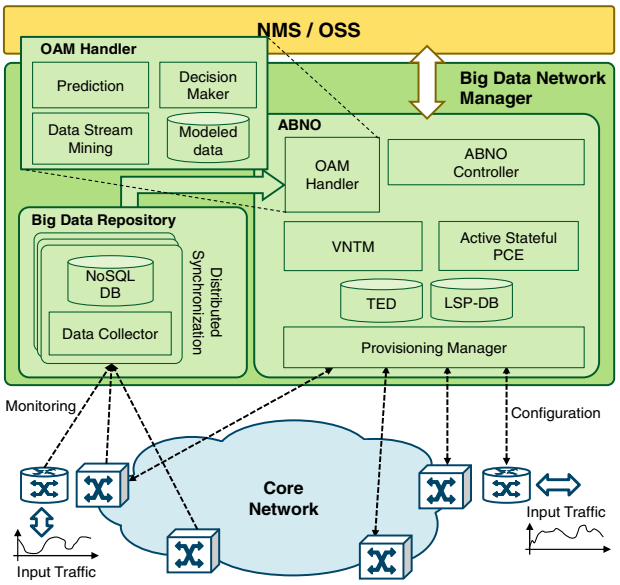

Fig. 6. Big data network manager architecture.

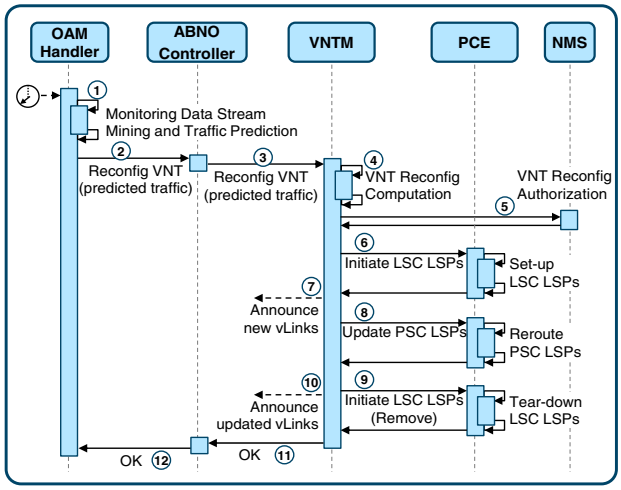

Fig. 7. Proposed workflow.

and the decision maker that decides whether the VNT should be updated based on the predicted traffic. Finally, the ABNO's VNTM is in charge of computing the new VNT.

Figure 7 presents the proposed workflow. Monitored traffic data is sent to the collected data repository ready to be analyzed. Periodically, ABNO's OAM handler retrieves aggregated monitored data and applies data stream mining techniques to the received data to transform monitored data into modeled data. Modeled data is used by the prediction module running the proposed ANN (labeled as 1 in Fig. 7). Based on the predicted traffic, a *decision-maker* module decides whether the VNT should be updated. In the case of VNT reconfiguration, the VNTM is in charge of computing the new VNT. To that end, the OAM handler issues a request to the ABNO controller that includes the predicted traffic together with some other parameters to facilitate VNTM computation (2), and the ABNO controller initiates the workflow forwarding the request to the VNTM (3).

The VNTM computes the new VNT with the predicted traffic matrix received from the OAM handler (4). Continuing with our seven-node VNT example, let us assume that the new VNT consists of adding the new virtual link 6–7 and reducing the capacity of some other vlinks. The solution is first notified to the NMS (5), and then its implementation is divided into a sequence to avoid traffic disruption as anticipated above: first, lightpath (LSC LSP) 6–7 is created (6) and the new vlink is advertised (7); next, PSC LSPs are rerouted (8) (a *make-before-brake* strategy to avoid disruption can be implemented) and unused capacity in vlinks 1-3 and 1-4 is removed by tearing down the underlying LSC LSPs (9); and the new vlinks' capacity is advertised (10). Upon VNT reconfiguration completion, the VNTM replies to the ABNO controller (11), which eventually replies to the OAM handler (12).

VI. Illustrative Results

This section focuses first on validating the VENTURE approach through simulations. Next, the proposed architecture is experimentally assessed in our SYNERGY testbed.

A. Simulation Results

For evaluation purposes, we implemented an event-driven simulator in OMNet++ containing the modules described in Fig. 4. To measure the effect of volumetric and directional changes in traffic, we implemented generators that inject traffic following two pre-defined traffic profiles called *Users* and Datacenter-to-Datacenter (*DC2DC*) [see average daily evolution in Fig. 8(a)]. In addition, some random values around the average value are usually observed in real traffic. In consequence, a function ε_t representing a random variable traffic is also added. We assume that ε_t follows a 0-centered normal (Gaussian) probability distribution, i.e., $\varepsilon_t \sim N(0, \sigma^2)$, where σ represents the standard deviation. In consequence, the daily traffic profile of every OD pair can be defined as $Y_{OD}(t) = \alpha \cdot f(t) + \varepsilon_t$, where function $f(t)$ represents the average traffic profile and α is a scaling factor in Mb/s.

Finally, the set of nodes was divided into two subsets to generate changes in the direction of the traffic; ODs with one of the nodes as their destination in the first subset follow the *Users* profile, while the others follow the *DC2DC* profile. We consider a scenario where a maximum of 26 × 100 Gb/s transponders per node are equipped. With such configuration, the static and threshold-based approaches are applied to a full-mesh 14-node VNT, where the initial capacity of each vlink ranges from 100 to 200 Gb/s.

The ANN models are trained by applying the fitting algorithm in Fig. 5 to a training dataset with modeled data belonging to the last week. The results in Fig. 8(b) illustrate the average size and goodness-of-fit of the ANN models. Recall that during the input selection phase, the number of inputs p is decreased, aiming at minimizing the AIC value. We observe that the minimum AIC is on average reached at $p = 4$, being mainly selected from those inputs from $t-1$ to $t-4$. The results from the hidden layer dimensioning phase are shown in the table embedded in Fig. 8(b) for a number of hidden neurons ranging from 1 to 3. Note that the minimum AIC is obtained for $s = 2$, which results in an ANN model with 10 coefficients that accurately predicts the output variable with a good

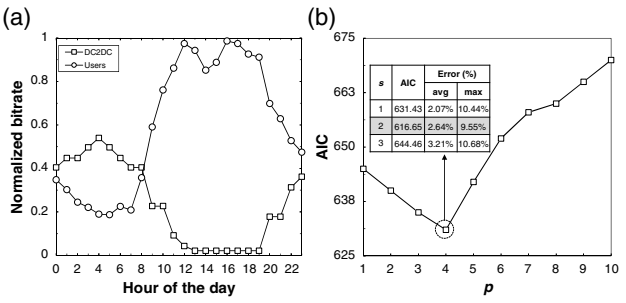

Fig. 8. (a) Average daily traffic profiles used in the simulation and (b) ANN goodness-of-fit.

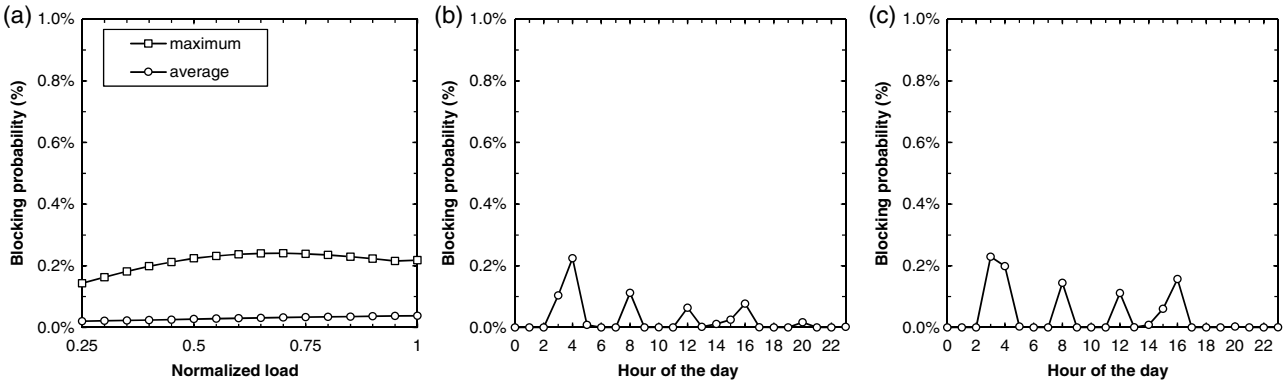

Fig. 9. (a) Average and maximum hourly blocking probability of VENTURE versus load. Blocking probability throughout one day and for normalized loads (b) 0.48 and (c) 1.0.

trade-off between the average and maximum relative errors (2.64% and 9.55%, respectively).

Next, we compare the effects on the unserved traffic and the number of used transponders under the threshold-based approach (we assumed 90% threshold) that runs continuously; under the VENTURE approach, it is triggered at fixed intervals of 1 h. For the sake of completeness, the static case where no reconfiguration is performed is also included.

Figure 9 presents the obtained blocking probability for the range of loads considered; values for both the static and the threshold-based approaches are omitted, since they yield a zero blocking probability. In the case of the VENTURE approach, Fig. 9(a) plots the average and maximum hourly blocking throughout a day. We observe that, for a wide range of traffic loads, the maximum blocking probability is below 0.24%, while on average it is virtually zero. Figures 9(b) and 9(c) analyze the evolution of the blocking probabilities during the day for the lowest and highest loads, respectively. We observe that small peaks of blocking probability appear related to abrupt changes in the injected traffic and last for a couple of hours at most, which is the time VENTURE takes to fully adapt the VNT to traffic changes with the specific configuration selected.

Figure 10 focuses on the use of transponders. Figure 10(a) plots the maximum transponder usage as a function of the load for each approach. Both, the static and the threshold-based approaches show a constant transponder usage for loads lower than 0.5, which is increased from that load up. For low loads, the capacity of vlinks in the fully meshed VNT is 100 Gb/s in both cases, and it is increased to 200 Gb/s for high loads under the static approach. The threshold-based approach, however, is able to manage the use of transponders by flexibly using available transponders to increment the capacity of vlinks running out of capacity; this way, it achieves transponder savings up to 11% with respect to the static VNT approach.

Interestingly, the transponder usage scales linearly with the load with VENTURE. Compared to the threshold-based approach, VENTURE obtains savings between 8% and 42%.

Figures 10(b) and 10(c) focus on the use of transponders along the day for the lowest and highest loads for the three approaches. Apart from the constant transponder usage in the static approach, we show the different usages of the threshold-based and the VENTURE approaches. In particular, we observe how the VENTURE approach is able to remarkably reduce up to 45% transponder usage at some hours, mainly when the *DC2DC* traffic profile is dominant. On the other hand, in those hours when *Users* traffic profiles dominate, transponder usage under VENTURE still outperforms that of the threshold-based approach.

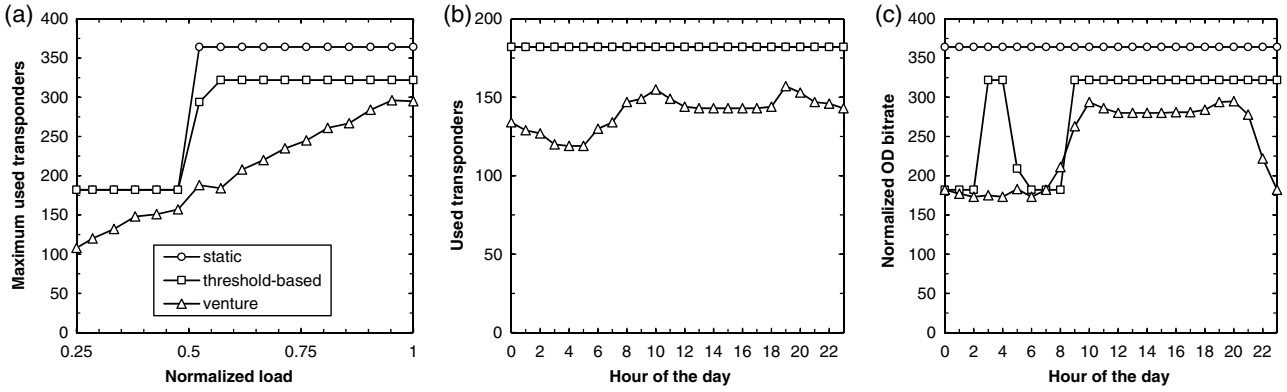

Fig. 10. (a) Maximum used transponders versus load. Used transponders along one day and for normalized loads (b) 0.48 and (c) 1.0.

In conclusion, the VENTURE approach maximizes the overall utilization of available transponders in two different ways: (i) by reconfiguring the virtual topology to follow traffic direction changes and (ii) by increasing the capacity of the vlinks when the traffic increases.

B. Experimental Assessment

Experiments have been carried out on the UPC's SYNERGY testbed. The Apache Cassandra database [14] was used as the big data repository, and a data collector module was implemented to offer a UDP-based interface to the monitors, storing the received data in Cassandra. Apache Spark [15] was used to implement data stream mining and machine learning techniques. Finally, the ABNO modules in Fig. 6 were implemented using UPC's iONE software [16]. An HTTP REST API interface was implemented between the OAM handler and the ABNO controller and from it to VNTM, so as to report the predicted traffic matrix. PCEP was used between the VNTM, PCE, and the provision manager. Finally, BGP-LS was used to synchronize TEDs. In particular, the VNTM is in charge of advertising topological changes in the VNT, including vlink creation and releasing, as well as updating vlink capacity changes.

Figure 11 illustrates monitored traffic data being periodically sent by the packet nodes to the data collector, as well as the request that the OAM handler issues to Cassandra's REST API to collect monitored data. UDP monitoring messages contain, among others, the source node and the timestamp of the sample, and for each aggregated flow leaving the node to a destination, its destination node and bitrate. After selecting and aggregating monitored data between the selected times t_i and t_j, Cassandra replies with a JSON-encoded matrix specifying, for each source–destination pair, the average, maximum, and minimum bitrates.

Figure 12 shows the meaningful messages exchanged between ABNO modules. For the sake of clarity, the messages are identified following the workflow in Fig. 6. The OAM handler sends a REST API request to the ABNO controller (message 2) containing the predicted traffic matrix for the next period. The details of that message are presented in Fig. 13. After receiving the predicted traffic matrix, the VNT computes the optimal VNT and issues requests to the PCE to implement the LSC LSPs supporting the new vlinks, reroute the selected PSC LSPs, and tear down unused LSC LSPs. In addition, VNT changes are advertised to the rest of the ABNO modules. The total process took 217 ms from the instant the OAM handler triggered the workflow.

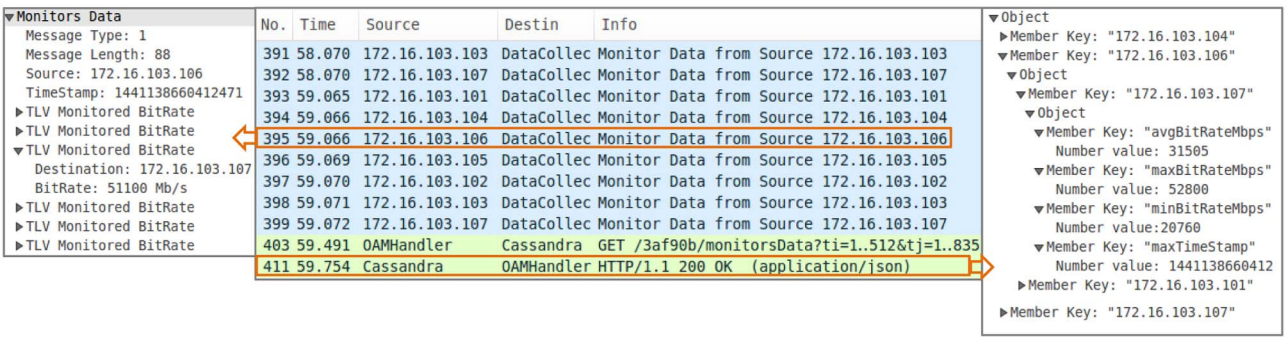

Fig. 11. Exchanged messages for monitored traffic collection.

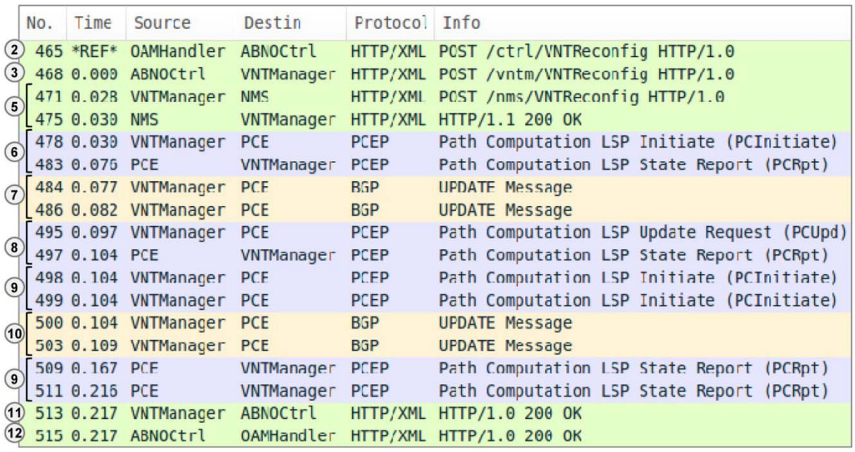

Fig. 12. Exchanged messages for VNT reconfiguration.

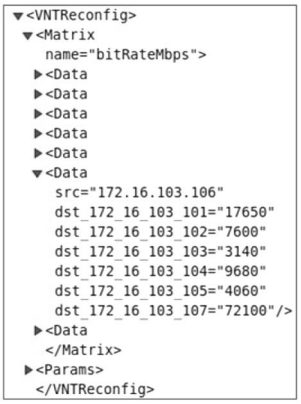

Fig. 13. Message 2 details.

VII. Conclusions

An efficient approach, called VENTURE, to adapt the current VNT to future traffic conditions aiming at minimizing TCO has been proposed. The approach consists of monitoring the OD traffic in IP/MPLS routers and applying data analytics to learn predictive models that are used as inputs of a reconfiguration problem. In particular, an ANN for every OD pair was proposed as a predictive model along with an algorithm to obtain a highly accurate ANN using as few coefficients as possible. The VENTURE reconfiguration problem was formally stated and formulated as an ILP. In view of its complexity for short-term valid solutions, a heuristic algorithm to provide near-optimal solutions in practical computation times was proposed.

In addition, a big data analytics OAM handler has been proposed to support VENTURE. Monitoring data is collected by the OAM handler and locally stored. Periodically, e.g., every hour, collected monitoring data are transformed into modeled data and the ANNs are used to predict the next-period traffic. A workflow is proposed, where the VNTM module solves the VENTURE reconfiguration problem to find the optimal VNT based on the predicted traffic computed by the OAM handler.

We compared the performance of VENTURE through simulation against the static and the threshold-based approaches. We observed savings between 8% and 42% in the number of transponders to be installed in the routers when the VENTURE approach was applied. In addition, VENTURE is able to deactivate transponders during low traffic hours, thus decreasing the energy consumption and releasing lightpaths from the underlying optical layer, which contributes to a cost reduction.

Finally, the proposed architecture was experimentally assessed in our SYNERGY testbed.

Acknowledgment

The research leading to these results received funding from the Spanish MINECO SYNERGY project (TEC2014-59995-R) and from the Catalan Institution for Research and Advanced Studies (ICREA). This work was presented in part at OFC 2016 [17,18].

References

[1] L. Velasco, L. M. Contreras, G. Ferraris, A. Stavdas, F. Cugini, M. Wiegand, and J. P. Fernández-Palacios, "A service-oriented hybrid access network and cloud architecture," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 159–165, 2015.

[2] M. Ruiz, M. Germán, L. M. Contreras, and L. Velasco, "Big data-backed video distribution in the telecom cloud," *Comput. Commun.*, vol. 84, pp. 1–11, 2016.

[3] D. King and A. Farrel, "A PCE-based architecture for application-based network operations," IETF RFC 7491, 2015.

[4] A. Aguado, M. Davis, S. Peng, M. V. Álvarez, V. López, T. Szyrkowiec, A. Autenrieth, R. Vilalta, A. Mayoral, R. Muñoz, R. Casellas, R. Martínez, N. Yoshikane, T. Tsuritani, R. Nejabati, and D. Simeonidou, "Dynamic virtual network reconfiguration over SDN orchestrated multi-technology optical transport domains," in *Proc. of the European Conf. on Optical Communications (ECOC)*, 2015.

[5] F. Agraz, L. Velasco, J. Perelló, M. Ruiz, S. Spadaro, G. Junyent, and J. Comellas, "Design and implementation of a GMPLS-controlled grooming-capable optical transport network," *J. Opt. Commun. Netw.*, vol. 1, pp. A258–A269, 2009.

[6] A. Gençata and B. Mukherjee, "Virtual-topology adaptation for WDM mesh networks under dynamic traffic," *IEEE Trans. Netw.*, vol. 11, pp. 236–247, 2003.

[7] Y. Ohsita, T. Miyamura, S. Arakawa, S. Ata, E. Oki, K. Shiomoto, and M. Murata, "Gradually reconfiguring VNT based on estimated traffic matrices," *IEEE Trans. Netw.*, vol. 18, pp. 177–189, 2010.

[8] N. Fernández, R. Durán, D. Siracusa, A. Francescon, I. de Miguel, E. Salvadori, J. Aguado, and R. Lorenzo, "Virtual topology reconfiguration in optical networks by means of cognition: Evaluation and experimental validation," *J. Opt. Commun. Netw.*, vol. 7, pp. A162–A173, 2015.

[9] L. Velasco, A. P. Vela, F. Morales, and M. Ruiz, "Designing, operating and re-optimizing elastic optical networks," *J. Lightwave Technol.*, to be published.

[10] J. Boyd, *Destruction and Creation*. U.S. Army Command and General Staff College, 1976.

[11] F. Emmert-Streib and M. Dehmer, *Information Theory and Statistical Learning*. New York: Springer, 2008.

[12] J. D. Hamilton, *Time Series Analysis*. New Jersey: Princeton University, 1994.

[13] K. Steiglitz and C. H. Papadimitriou, *Combinatorial Optimization: Algorithms and Complexity*. New Jersey: Prentice Hall, 1982.

[14] Apache Cassandra [Online]. Available: http://cassandra.apache.org/.

[15] Apache Spark [Online]. Available: http://spark.apache.org/.

[16] L. Velasco and L. Gifre, "iONE: A workflow-oriented ABNO implementation," in *Proc. of the Photonics in Switching Conf.*, 2015.

[17] L. Gifre, L. M. Contreras, V. Lopez, and L. Velasco, "Big data analytics in support of virtual network topology adaptability," in *Optical Fiber Communication Conf. (OFC)*, 2016.

[18] F. Morales, M. Ruiz, and L. Velasco, "Virtual network topology reconfiguration based on big data analytics for traffic prediction," in *Optical Fiber Communication Conf. (OFC)*, 2016.