Dynamic Core VNT Adaptability based on Predictive Metro-Flow Traffic Models

F. Morales, Ll. Gifre, F. Paolucci, M. Ruiz, F. Cugini, P. Castoldi, and L. Velasco

Abstract—**MPLS-over-optical virtual network topologies (VNT) can be adapted to near future traffic matrices based on predictive models that are estimated by applying data analytics on monitored origin-destination (OD) traffic. However, the deployment of independent SDN controllers for core and metro segments can bring large inefficiencies to this core network reconfiguration based on traffic prediction when traffic flows from metro areas are rerouted to different ingress nodes in the core. In such case, OD traffic patterns in the core might severely change thus affecting the quality of the predictive OD models. New traffic models' re-estimation usually entails long time during which no predictive capabilities are available for the network operator. To alleviate this problem, we propose to extend data analytics to metro networks to obtain predictive models for the metro-flows; by knowing how these flows are aggregated into OD pairs in the core, we can also aggregate their predictive models thus accurately predicting OD traffic and therefore, enabling core VNT reconfiguration. To obtain quality metro-flow models, we propose an estimation algorithm that processes monitored data and returns a predictive model. In addition, a Flow Controller is proposed for the control architecture to allow metro and core controllers to exchange metro-flow model information. The proposed model aggregation is evaluated through exhaustive simulation, and eventually experimentally assessed together with the Flow Controller in a testbed connecting premises in CNIT (Pisa, Italy) and UPC (Barcelona, Spain).**

Index Terms—**Predictive traffic modelling, Traffic model aggregation.**

I. Introduction

Many connectivity services require considerably less capacity than the one that optical connections offer; this forces operators to keep differentiated metro and core network segments and to deploy multilayer networks in the core to perform the required grooming/aggregation functionality [3]; metro areas are usually connected to the core through two or more nodes for redundancy and load balancing purposes.

To properly design and dimension the core MPLS-over-optical virtual network topology (VNT) during the network planning phase, traffic matrices containing maximum traffic values between every origin-destination (OD) core pairs are commonly used as input of planning problems. However, that approach entails large overprovisioning, especially in Telecom Cloud scenarios [4] where traffic largely varies along the day in terms of volume and directionality; this, remarkably increases Capital Expenditures (CAPEX). In addition, independent software-defined networking (SDN) controllers are commonly deployed for the different segments (metro and core) and technologies (MPLS and optical). This domain-managed network scenario might lead to local optimal resource allocation since flows in metro-areas (*metro-flows*) are routed considering only resource availability in the metro. As a result, the core VNT might become congested if metro-flows are rerouted and enter in the core through a different node because of metro-scope re-optimization.

To support and automate VNT adaptability, authors in [5] proposed a data analytics -enabled network manager architecture to store OD traffic monitoring data featuring prediction and detection of traffic anomalies [6] and network reconfiguration in response [7]. Predictive traffic models can be obtained by applying machine learning to monitored traffic data; e.g., artificial neural networks (ANN) were considered in [5]. Those predictive traffic models can be used in the process of VNT reconfiguration, where the VNT can be proactively adapted for the predicted near future traffic conditions; this improves resource utilization and reduces overprovisioning. In particular, the VENTURE problem was proposed in [5] to periodically adapt the VNT to current and predicted traffic.

However, the efficiency of that proposal strongly depends on accurate predictions, which entails long monitoring data collection times (e.g., several weeks). In addition, OD traffic patterns are likely to evolve in time (generally referred to as *concept drift*), so the accuracy of current models needs to be monitored to eventually trigger a re-estimation if their quality drops below the desired level. In this regard, several works focusing on concept drift detection and adaption have been studied in the literature (see a survey in [8]).

In the case of long-term changes, predictive models remain usefull even though a re-estimation has been triggered. Nonetheless, a sudden or abrupt OD traffic (core-flows) change as a result of metro-flow rerouting makes the related predictive models become inmediately obsolete and cannot be used any more. Consequently, VNT adaptability becomes completely reactive as a consequence of the lack of prediction, until new predictive models are computed based on new monitored data several days or even weeks later. Depending on the frequency of such reroutings, mantaning accurate traffic models by means of re-estimation might not be possible due to too short model

Manuscript received July 8, 2017.

This work was presented in part at OFC 2017 [1] and in ECOC 2017 [2].

Fernando Morales, Marc Ruiz and Luis Velasco (lvelasco@ac.upc.edu) are with the optical communications group (GCO) at Universitat Poltècnica de Catalunya (UPC), Barcelona, Spain. Lluís Gifre is with Universidad Autónoma de Madrid (UAM), Madrid, Spain. Francesco Paolucci and Piero Castoldi are with Scuola Superiore Sant'Anna, Pisa, Italy. Filippo Cugini is with CNIT, Pisa, Italy.

training periods.

In view of this, new mechanisms need to be devised to keep predictive capabilities in the core network in front of changing core-flow traffic patterns because of metro-flow rerouting. In this regard, some authors have proposed to monitor (small) individual flows instead of the (large) aggregated flows for traffic modelling purposes. For instance, authors in [9] proposed a machine learning procedure that intelligently de-aggregates relevant monitored traffic flows and aggregates the rest to achieve accurate traffic estimations. However, this kind of selective flow monitoring approaches lacks the flexibility required to adapt OD traffic models against any potential metro-flow rerouting.

Another option is to apply data analytics in the metro networks, based on monitoring data from metro nodes, to obtain estimation models of metro-flow traffic. Metro-flow predictive models can be used to re-estimate obsolete OD models in the core. For instance, authors in [10] proposed AutoRegressive Integrated Moving Average (ARIMA) for modelling and forecasting metro area network traffic flows as a function of application-layer traffic flow models in IP networks. However, the lack of collaborative control architectures prevents from exchanging metro-flow models between the core and the metro controllers.

It is worth mentioning that the accuracy of predictive OD models cannot be based only on extensive metro-flow traffic monitoring. Although the literature offers a broad range of mathematical models for fitting flow traffic data [11], techniques for aggregatting metro-flow models to guarantee reliable and accurate OD traffic predictions need to be explored. Among related works, the survey presented in [12] reviews different approaches to predict time series of an aggregate by means of each individual flow forecast.

In this paper, we extend our previous work in [1] and [2] and propose composing OD traffic models based on the aggregation of metro-flow traffic models. Section II presents the drawbacks of using predictive OD traffic models based on monitoring the core, motivating the need of extending data analytics to the metro areas to obtain metro-flow predictive models that can be afterwards aggregated to obtain updated OD models in practical times. Based on aggregated models, we can predict future traffic for the OD pairs and use it as input for core VNT re-optimization purposes. In addition, we aim at requiring low storage and computing requirements to bring data analytics near the network nodes, thus opening the possibility of applying decentralized data analytics [6]. Specifically, the contribution of this paper is three-fold:

1) In Section III, metro-flow traffic modelling is first introduced and two algorithms are presented for *i*) obtaining metro-flow predictive models from monitoring data and *ii*) for evaluating these models to obtain traffic predictions. We devise models with low storage and computing requirements to enable their utilization near the network nodes.
2) In addition, we formally present the aggregation of metro-flow models into OD models and the algorithm responsible for obtaining those models, either by updating or building obsolete OD models in the event of metro-flow rerouting.
3) In Section IV, we propose a Flow Controller where data regarding metro-flows is stored and can be queried by the controllers. Metro controllers estimate traffic models based on monitoring metro-flow traffic and those models are available in the Flow Controller for the core controller to access them. In addition, metro controllers announce metro-flows rerouting to the core controller.

The discussion is supported by the results presented in Section V. The accuracy of both metro-flow and OD predictive models is validated by means of exhaustive numerical results, and their validity for supporting core VNT reconfiguration based on traffic prediction under metro-flow rerouting is evaluated in a network simulation scenario. Finally, the proposed Flow Controller to support traffic aggregation is experimentally assessed.

II. Core OD Traffic Prediction

Let us assume the scenario in Fig. 1, where a core MPLS-over-optical VNT interconnects several metro networks using dual homing. Although traffic flows can be monitored at any intermediate node, let us assume that metro-flows are monitored in metro nodes, while core OD traffic is obviously monitored in core nodes. Monitored traffic is sent at regular intervals to the corresponding controller, where predictive models can be estimated once enough monitoring data has been collected. Once these models are available, the VENTURE problem is executed at regular intervals (e.g. hourly) to adapt the VNT to the future traffic conditions [5].

Modelling core OD and metro-flow traffic independently at each network segment can lead to a degraded performance in the quality of the predictive models after metro-flow rerouting. OD pairs might aggregate many metro-flows and hence, rerouting some of the metro-flows in the metro areas might change the aggregated traffic pattern of some ODs in the core network. For illustrative purposes, Fig. 1 presents an example of a metro-flow (*mf1*) originated at some metro network and routed towards datacenter DC2 through the core VNT. The metro-flow is encapsulated in a MPLS Label Switched Path (LSP) and routed through the VNT. Initially, the LSP enters the core VNT through ingress node R2 towards egress node

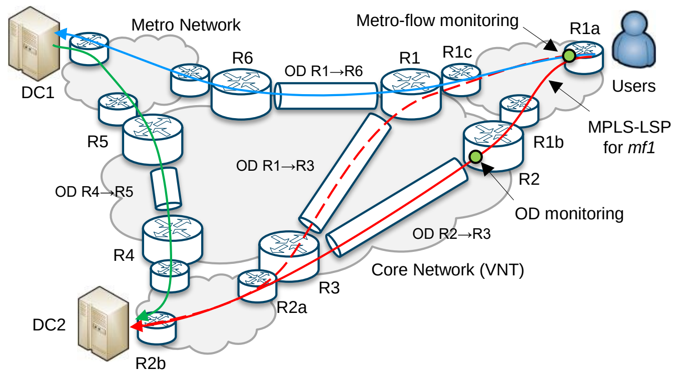


Fig. 1. OD traffic can change due to metro-wide re-configuration.

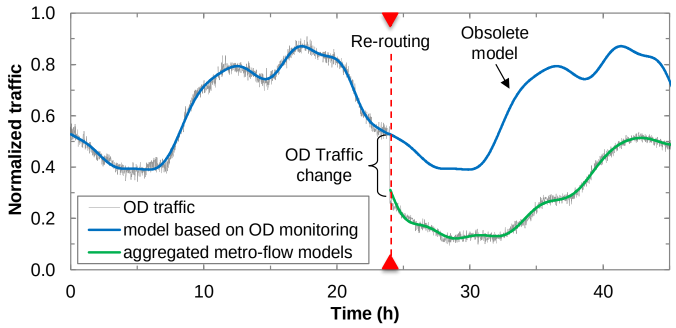


Fig. 2. Example of a traffic model that becomes obsolete.

R3, so it is aggregated into core OD pair R2->R3. Due to metro re-optimization, the metro controller reroutes the metro-flow LSP so that to enter the core VNT through edge node R1, being aggregated into core OD pair R1->R3. As a result, the traffic profiles of both OD pairs (R1->R3 and R2->R3) have now changed.

Fig. 2 illustrates the traffic profile change in OD R2->R3. Before the rerouting event, the predictive core OD model perfectly fits the actual traffic; once metro-flow LSP *mf1* has been rerouted, the OD traffic profile changes and the corresponding predictive model becomes obsolete, thus triggering a re-estimation based on new monitoring data. Note that the difference between the actual traffic volume and the obsolete prediction can be mistakenly confused with an OD traffic anomaly [6], which would trigger unnecessary network reconfiguration. During this re-estimation process, the VENTURE algorithm will not be available thus allowing only reactive approaches to reconfigure the VNT.

Alternatively, OD traffic can be predicted by considering its relationship with metro-flow traffic. Effectively, by aggregating the traffic models of those metro-flows being routed through each core OD pair we can produce new, updated OD traffic models. Immediately after the rerouting event, the obsolete model for ODs R1->R3 and R2->R3 are replaced by new ones based on the aggregation of metro-flows predictive models. Fig. 2 shows how the aggregation of metro-flow traffic predictions perfectly fits the new OD traffic pattern without the need of restarting the process of monitoring and estimation from scratch. By following this approach, the core network operator can keep the predictive capabilities, provided that some sort of coordination between metro and core segments exist.

III. Traffic Modeling and Model Aggregation

This section is devoted to traffic modelling, as well as to provide detailed procedures for traffic prediction using such models. The methodology is general, so it can be used for estimating models for both metro-flows and core OD traffic. Assuming that the methodology is used to estimate models for metro-flow traffic, a procedure is then proposed for aggregating those models to either build new OD models or keep them updated after metro-flow rerouting. An evaluation algorithm for their utilization in VNT reconfiguration is eventually presented.

A. Traffic modeling

Let us consider that, after a certain time of monitoring activity, a time series of monitored traffic samples Y is available for model estimation. Each sample $y \in Y$ is obtained following the architecture in [13]: initially, bitrate is monitored at packet nodes where counters are continuously updated during short *granularity periods* of duration G (e.g., every 1 minute). Once a period ends, bit counters are processed to produce a new bitrate measurement. In order to reduce the amount of data sent to the centralized controller, measures are aggregated in the node computing the arithmetic mean according to a larger *monitoring period* T (e.g., every 15 minutes) to produce a monitoring sample y=<*time*, *bitrate*>.

By modeling a traffic flow, we aim at obtaining an estimation of the expected mean (or average) bitrate μ and the variance σ^2 as a function of time. The proposed estimation algorithm is presented in Table I. It receives a time series Y for a given traffic flow and the monitoring period T and returns predictive models based on this input data. Specifically, two models (for μ and σ^2) consisting in two piece-wise linear functions of a certain number of segments are computed. The main variables used in the algorithm are:

perDur	duration of the period.
perStart	period starting time.
nSegm	number of segments of the piece-wise linear functions.
segmLength	length of each segment.

The first part of the algorithm (lines 1-5 in Table I) is a pre-processing phase consisting in grouping traffic values in time series Y by its relative time within the longest identified period. A seasonality detection procedure on the time series [14] is applied to compute the most likely period in the data (line 1). The detection procedure consists in computing the autocorrelation function to detect the distance between consecutive correlation peaks, identified as the duration of the period; *perStart* is set in consequence, e.g., 00:00h if a daily

Table I Traffic Model Estimation Algorithm

```
INPUT Y, T
OUTPUT model
1:  <perStart, perDur> ← identifyPeriod (Y)
2:  X ← []; nSegm ← ⌈ perDur / T ⌉
3:  for y = <time,bitrate> in Y do
4:      t ← ⌊ ((y.time - perStart) % perDur) / T ⌋
5:      push(y.bitrate, X{t})
6:  U ← []; V ← []; μ ← ∅; σ² ← ∅
```

```
 7: segmLength ← perDur / nSegm
 8: for t in 0..nSegm do
 9:   if t < nSegm then
10:     U[t] ← compute mean u(X{t}) (eq. (1))
11:     V[t] ← compute variance v(X{t}) (eq. (2))
12:   if t = 0 continue
13:   if t < nSegm then t' ← t – 1 else t ← 0; t'← nSegm - 1
14:   a_μ ← U[t]; a_σ2 ← V[t]
15:   b_μ ← compute slope b(U[t'], U[t], segmLength) (eq. (3))
16:   b_σ2 ← compute slope b(V[t'], V[t], segmLength) (eq. (3))
17:   μ[t] ← <a_μ, b_μ>; σ²[t] ← <a_σ2, b_σ2>
18: return model=<perStart, perDur, nSegm, μ, σ²>
```

period is detected. Since samples are monitored at regular intervals, *nSegm* can be easily computed from *perDur* and *T* (line 2). Once the period has been obtained, data is grouped by segments, i.e., expressed with a time *t* relative to the period. To this aim, every sample in *Y* is retrieved, its relative time *t* computed and the traffic value pushed to the vector that contains all the samples collected at *t*, which is stored in data set *X* (lines 3-5).

After the pre-processing phase, *X* contains the same data in *Y* but properly grouped to easily compute the coefficients of the linear equation of each segment of the piece-wise linear functions. Every vector *X*{*t*} is used to compute two consecutive linear equations: for segment [*t*-1, *t*] it is used as ending edge whereas for segment [*t*, *t*+1] it is used as starting edge. Thus, for each edge of a segment, the empirical mean and variance are computed according to those typical moment estimators [15] detailed in eq. (1) and (2) (lines 6-11).

$$u(X)=\frac{1}{|X|}\sum_{i=0}^{|X|-1}X[i] \tag{1}$$

$$v(X)=\frac{\sum_{i=0}^{|X|-1}\left(X[i]-u(X)\right)^2}{|X|-1} \tag{2}$$

As soon as the empirical mean and variance are available for both edges of a segment, the linear equations (intercept and slope) of that segment for both μ and σ^2 models are computed (lines 12-17). Segments are identified by an intercept *a* that equals the empirical value at the starting of the segment and a slope *b* computed from the values at both edges and the segment length according to equation (3). Finally, the model consisting in the obtained period, the number of segments, and the piece-wise linear functions μ and σ^2, is returned (line 18). In practice, *nSegm* is implicitly stored as the size of μ and σ^2.

$$b(x_i,x_j,l)=\frac{x_j-x_i}{l} \tag{3}$$

Once traffic models are available, they can be used to predict future traffic. Table II shows the evaluation algorithm that receives a predictive model and the absolute time for which a prediction is needed. First, the absolute time needs to be transformed to a time (*t*) within the model's period (lines 1-2). Next, the linear equations (slope and intercept) for μ and σ^2 that enclose *t* are selected (lines 3-6) and finally evaluated (lines 7-8). The algorithm returns the average bitrate and the variance prediction for the requested time (line 9). The number of operations that the algorithm executes is constant and does not depend on the size of the input; therefore, its time complexity is constant, which translates in fast evaluation times in practice. In addition, it requires a constant amount of additional memory apart from that used to load the model (space complexity is linear). This facilitates its utilization in computational resource scarcity environments. The storage requirements of these models are studied in Section V for their use in the network nodes.

TABLE II MODEL EVALUATION ALGORITHM

```
INPUT model, time
OUTPUT <μ, σ²>
1: segmLength ← model.perDur / model.nSegm
2: t ← (time – model.perStart) % model.perDur
3: s ← ⌊ t / segmLength ⌋
4: offset ← t % segmLength
5: [a_μ, b_μ] ← model.μ[s]
6: [a_σ2, b_σ2] ← model.σ²[s]
7: μ ← a_μ, + b_μ * offset
8: σ² ← a_σ2 + b_σ2 * offset
9: return <μ, σ²>
```

The granularity of the previous prediction is implicitly given by the monitoring period *T*, which might not be fine enough for some algorithms, such as anomaly detection [6]. This specially affects the σ^2 model, since the variance of bitrate at a granularity smaller than *T* (e.g., *G*) tends to be higher. For illustrative purposes, let us imagine that μ and σ^2 models have been obtained setting a granularity *G*=1 min and a monitoring period *T*=15 min. As a result of the implicit traffic aggregation induced by *T*, we will obtain: *i*) a μ model estimation close to the mean traffic observed at a granularity *G* and *ii*) a σ^2 model estimation significantly smaller than that observed at a granularity *G*. Thus, if we want to use the σ^2 model to predict traffic with granularity 1 minute its accuracy needs to be improved. To that end, we propose to use approximated predictions applying corrections derived from theory of estimation in statistics [15].

B. OD pair traffic modelling

Let us consider now a particular core OD pair *od* and its set of aggregated metro-flows *F*(*od*), where predictive models for the mean (μ_f) and the variance (σ^2_f) are available for each metro-flow *f* in *F*(*od*). From the linearity of the expectation [15], the average OD traffic (μ_{od}) is equivalent to the summation of the metro-flow average traffic (eq. (4)). Regarding the OD pair variance (σ^2_{od}), it can be expressed as the summation of metro-flow variances if and only if variances are uncorrelated (eq. (5)). Correlation is commonly observed in the traffic and has been already studied in the literature [16]. Therefore, it would not be realistic to assume that the aggregated metro-flows have uncorrelated traffic if, for instance, they convey similar service traffic. When correlation is present between metro-flows, the expression of the OD variance becomes more complex since it additional nonzero covariances between all pairs of aggregated flows needs to be added [15]. In section V we analyze the bias introduced in the estimation of σ^2_{od} when covariance terms are excluded (linear aggregation).

$$\mu_{od}(t) = \sum_{f \in F(od)} \mu_f(t) \quad (4)$$

$$\sigma^2_{od}(t) = \sum_{f \in F(od)} \sigma^2_f(t) \quad (5)$$

Table III presents the proposed algorithm to create or update core OD traffic models after a metro-flow rerouting event. It receives the set *Q* with all OD pairs, where each pair includes its model *m* and the set of aggregated metro-flows.

TABLE III OD MODEL UPDATE ALGORITHM

```
INPUT: $Q = \{<od, m, F(od)>\}$
OUTPUT: $S = \{<od, m'>\}$
1:  $Q_{obs}$ ← getObsoleteModels($Q$)
2:  $S \leftarrow \emptyset$
3:  for each $q = <od, m, F(od)>$ in $Q_{obs}$ do
4:      if type($m$) = NEW_CORE then
5:          $m' \leftarrow m$
6:      else if type($m$) = NEW_METRO then
7:          $m'$ ← newAggregate($F(od)$) (eqs. (4),(5))
8:      else if type($m$) = UPDATE then
9:          $m'$ ← updateAggregate($m$, $F(od)$) (eqs.
10:         (6),(7))
11:     $S \leftarrow S \cup \{<od, m'>\}$
    return $S$
```

First, the set of obsolete models is found by inspecting the current aggregation of the metro-flows (line 1 in Table III). For each obsolete OD model, the type of the model determines whether it is a model estimated from core traffic monitoring (NEW_CORE) (lines 4-5), from metro traffic monitoring (NEW_METRO) (lines 6-7) or it is model that needs to be updated (UPDATE) by including the new metro-flows entering the OD pair and excluding those ones leaving it from the prediction (lines 8-9). For the updating process, we can take advantage of the linearity of the mean and the variance in the aggregation to produce updates applying equations (6) and (7), only taking into account those metro-flows leaving and entering the core OD.

$$\mu_{od}(t) += \sum_{f \in F_{IN}(od)} \mu_f(t) - \sum_{f \in F_{OUT}(od)} \mu_f(t) \quad (6)$$

$$\sigma^2_{od}(t) += \sum_{f \in F_{IN}(od)} \sigma^2_f(t) - \sum_{F_{OUT}(od)} \sigma^2_f(t) \quad (7)$$

Finally, the algorithm returns the set of updated models (line 9). Note that the aggregation of μ and σ^2 models entails adding the metro-flows piece-wise linear functions. However, this is immediate from the piece-wise linearity of these functions by simply adding the slopes and intercepts for each segment to obtain the aggregated piece-wise linear function for μ_{od} and σ^2_{od}. For the sake of simplicity, we assume that all aggregated models present the same period and number of segments. Otherwise, additional computation would be required to obtain the partitioning resulting from merging all the piecewise linear functions μ_f and σ^2_f using the least common multiple period.

Let us now analyze the worst-case time complexity of the previous algorithm, assuming that all OD models need to be re-estimated ($|Q_{obs}|=|Q|$) with a maximum number of metro-flows $|F|$ for each re-estimation. Let us also assume that all aggregated models are of type NEW_METRO (i.e., built from scratch using eqs. (4) and (5)) being this the most time-consuming case. Then, the worst-time complexity is $O(|Q|\cdot|F|\cdot nSegm)$.

Limiting the model evaluation to the mean and the variance as presented in Table II discards other interesting estimations such as the maximum bitrate, important to re-optimize the VNT [5]. Although this estimation is not directly provided by the algorithm, we can obtain it in a later stage by applying results from probability theory involving μ and σ^2. Given a time *t*, let us assume that the traffic is distributed following a normal distribution $N(\mu, \sigma^2)$. Then, eq. (8) predicts the maximum bitrate with a confidence of 95% and 99.7%, respectively for k=2 and k=3 [17].

$$\max_{od}(t) \approx \mu_{od}(t) + k\sqrt{\sigma^2_{od}(t)} \quad (8)$$

Finally, note that the previous prediction provides the maximum traffic with granularity *T* (e.g., 15 min), which might not be enough for VNT reconfiguration actions typically requiring the maximum predicted bitrate during larger intervals (e.g., one hour [5]). One solution to obtain predictions for larger intervals is to produce several predictions along the considered interval and keep the maximum value obtained. Although this procedure entails multiple evaluations of the model thus, increasing the complexity of the proposed algorithm, the number of these evaluations needed to ensure the highest prediction provided by the model is known. This follows from the fact that eq. (8) defines a piece-wise linear function for the maximum prediction and that the maximum value in a piece-wise linear function segment takes place at the edges. Thus, it is enough to produce predictions at those time points where two consecutive segments of the piece-wise linear functions are connected. Note that the presented approach assumes that OD traffic can be accurately approximated as the sum of the metro-flow bitrate participating in the OD. In Section V, this assumption is validated for a wide range of traffic conditions.

IV. PROPOSED CONTROL ARCHITECTURE AND WORKFLOWS

The previous section showed how metro-flow predictive traffic models can be aggregated to obtain core OD traffic models. Therefore, metro controllers need to share information about metro-flow models with the core controller for the latter to compute predictive traffic models for each core OD pair.

Fig. 3 presents the proposed architecture that allows this exchange of information. In the architecture, we assume that every controller includes a data analytics module capable of collating monitoring data form the nodes; the IPFIX protocol [18] can be used to that end. The data analytics module processes monitored data and estimates traffic predictive models [13]. The metro controller includes also an SDN controller responsible for the configuration of the network devices (using e.g., OpenFlow), while the core controller includes an additional database to store the metro-flow -related data; a Stateful Path Computation Element (PCE), with the Traffic Engineering Database (TED) and the LSP-Database (DB), complete the core controller's architecture.

A new centralized system, named as *Flow Controller*, contains a repository to store metro-flow-related data that metro controllers update and the core controller interrogates to produce predictive OD traffic models. The repository stores for each metro-flow: *i*) its LSP's symbolic path name; *ii*) the border metro nodes through which the flow leaves and enters different metro areas. For instance, the output and input metro nodes for metro-flow *mf1* in Fig. 3 are R1b and R2a, respectively; and *iii*) the metro-flow predictive model that includes its period (starting time and duration) as well as μ_f and σ^2_f piecewise linear functions.

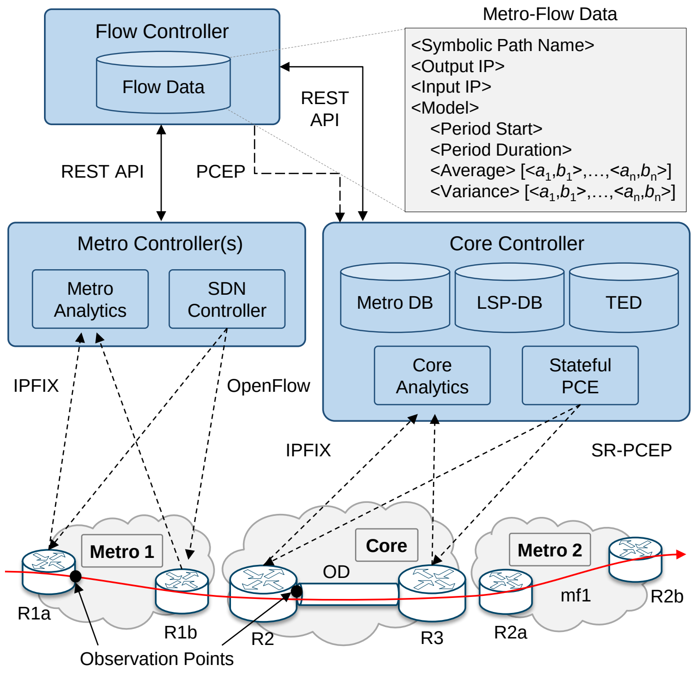


Fig. 3. Proposed network architecture.

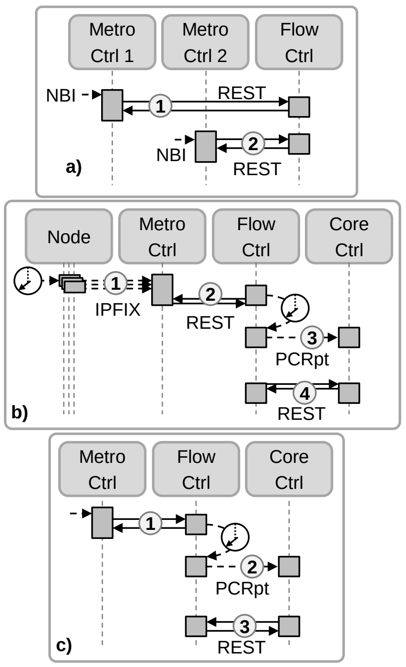


Fig. 4. Metro-flow set-up (a), Metro-flow model estimation (b) and Metro-flow rerouting (c).

Three workflows are defined to store/retrieve metro-flow data in/from the Flow Controller. Fig. 4a shows the first workflow triggered when a new LSP for a metro-flow is set-up across different metro domains; every involved metro-controller sends a JSON-encoded REST API message to the Flow Controller with the LSP symbolic path name and the ingress or egress metro border node. The Flow Controller creates a new entry in the flow repository and populates it correlating data received from different metro controllers.

Once the LSP for the metro-flow is operational, its traffic is monitored in any of the metro nodes and monitored data is exported by means of IPFIX messages (message 1 in Fig. 4b) to the corresponding metro controller. When enough monitoring data has been collected, the analytics module in the metro controller estimates a new traffic model for the metro-flow, or it re-estimates an existing one, and it makes available the model in the Flow Controller by means of a REST API message that includes the LSP's symbolic path name for identification (message 2).

After a metro-flow data entry has been completed or updated, the Flow Controller notifies the core controller by issuing a PCEP PCReport message [19] (message 3) containing the list of updated metro-flow LSPs (a delay has been introduced to prevent flooding the core controller with several updates from different metro controllers). The core controller can now retrieve updated metro-flow data by issuing a REST API request (message 4), being thus able to update obsolete OD traffic models. When a metro controller decides to reroute one or more metro-flows, it updates the metro-flow data in the Flow Controller (message 1 in Fig. 4c), which proceeds as described for Fig. 4b.

V. Results

In this section, we first evaluate the proposed metro-flow traffic modelling procedure by means of realistic synthetic traffic data. Once validated, we analyze the key aspects regarding metro-flow model aggregation into core OD models. After validating the proposed modelling procedure, we will illustrate the applicability of the proposed OD models for successfully solving the VENTURE problem. Finally, the proposed Flow Controller architecture supporting this traffic modeling approach is experimentally assessed.

A. *Metro-flow traffic modelling analysis*

Due to the diversity of service types that can be conveyed in realistic metro-flows, we generated traffic according to two clearly differentiated profiles. The first traffic profile, named as *Users*, represents the traffic aggregation of hundreds of end users consuming high-bandwidth applications such as video-on-demand or live TV, with higher activity at evening hours, i.e. prime-time [20]. The second traffic profile, named as *Datacenter (DC)*, aggregates traffic of tens of DC to DC connectivity services required for dynamic management activities of distributed DCs, such as DB synchronization or VM migration [21]. The daily traffic pattern of these two profiles is illustrated in Fig. 5. Expected traffic pattern of metro-flows will follow one of the profiles, where synthetic monitoring samples will be generated considering the value given by the profile plus a random fluctuation according to a variance which magnitude is proportional to the profile.

Once several months of monitoring data at a granularity G=1 min are generated, we first focus on analyzing the quality of μ model estimation under different monitoring periods T of 1, 5, 15, 30 and 60 minutes. For each T, we retrieved training time series Y containing several months of monitored data used as input to the estimator algorithm in Table I. To check the quality of the estimation, we compared it against a new, validation time series Y^* for each traffic profile; in particular, for each data value $y \in Y^*$ the error of the prediction was computed as the relative difference between the prediction and the real monitored value y. The average and maximum error were then compared for different values of T. By observing the average error, we find values under 1% for both traffic profiles and for T lower than 30 min. However, when looking at the maximum error (i.e., the worst prediction) we notice important differences between both traffic profiles. Whilst the *Users* profile yields maximum error values below 2% for all T, those for the *DC* profile remain low only when $T \leq 15$ min, while exceeding 10% for larger T; this is caused by the combination of abrupt changes in its daily profile and the loss of information because of aggregation. Consequently, setting T = 15 min provides a good tradeoff between information loss and prediction quality and hence, we fix this value for the rest of the study. Similar experiments were conducted to evaluate the estimation of the variance σ^2. Fig. 6a shows the error resulting from comparing the estimation of σ^2 against the variance used to generate the training time series. It can be observed that the estimation offers a 0-centered error bounded by ±10%. Results are shown averaged for the *Users* and *DC* profiles, yielding similar results for both individually.

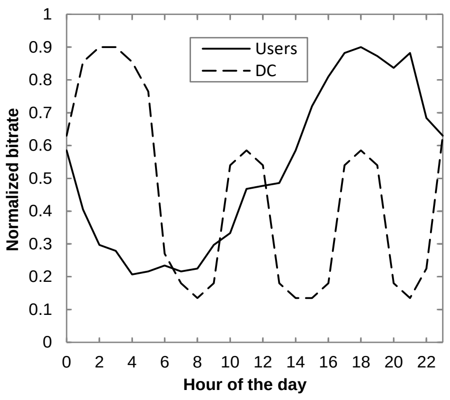


Fig. 5. *Users* and *DC* metro-flow traffic profiles.

Let us now analyze the impact of the amount of training data (i.e., $|Y|$) in the quality of the predictive models. Monitoring traffic during the right period of time is crucial to produce quality models whilst minimizing the time for new model availability. To evaluate this, we conducted experiments where μ and σ^2 are estimated and evaluated varying $|Y|$ between 2 days and 3 months.

Fig. 6b shows the maximum error for μ and σ^2 estimations for different values of $|Y|$. Values are normalized to the ones obtained with $|Y|$=3 months, which offered an acceptable error. Although μ can be estimated with less than 5% maximum error in about 10 days, a maximum error of 60% is obtained for σ^2 for the same time. To decrease the maximum error $|Y|$ need to be increased up to 2 months to keep maximum prediction errors under 20%. In this work, we consider $|Y|$=3 months of traffic monitoring to train models to accurately fit the behavior of metro-flows. Fig. 7 shows one day of monitoring traffic data, as well as the prediction of the μ model (red line) and the confidence interval at 95% that is obtained by means of the σ^2 model (dashed lines).

Finally, let us analyze the storage requirements for the proposed traffic models. Each traffic model requires $2 \cdot (2N+1)$ floating point numbers to be stored, where N is the number of model coefficients. As an example, in a 100-node network each node would require 153 KB to store all the OD models originated in the node modeled with daily periodicity, being this value increased to 4 MB if monthly periodicity is used.

B. Metro-flow model aggregation analysis

Let us now analyze under which traffic conditions the proposed predictive model aggregation (formally stated by eqs. (4) and (5)) is valid. These equations enatil that the bitrate of the OD pair resulting from the metro-flow aggregation accurately approximates the addition of metro-flow bitrates. Therefore, the analysis focuses on the traffic conditions that allow such approximation.

The scenario were several metro-flows are aggregated into a single core OD pair can be modelled using queuing theory; metro-flow packets arrive to a packet-switching node, where they are queued and aggregated into a single OD pair at a maximum give rate, e.g., 100 Gb/s. As proposed in [22], this can be mathematically modeled by a G/D/1 queue, assuming a generic distribution of arriving packets and a single server with constant service time. In such queue, the sum of the metro-flow bitrates accurately approximates the resulting OD pair bitrate as long as the actual service rate does not exceed 90% of its maximum (queue length remains small) [23]. Such condition must be ensured during the process of VNT reconfiguration, and therefore vlink capacities must be dimensioned to ensure that OD capacity utilization is under the aforementioned threshold.

Let us finally evaluate the bias introduced in the estimation of σ^2_{od} (as presented in section III.B) by running experiments where the maximum bitrate (eq. (8)) is predicted for a single OD pair aggregating *Users* metro-flows only, which leads to a large and positive covariance. Fig. 8a shows the minimum value of k needed to predict the maximum bitrate below a given error for different number of aggregated metro-flows. Although the typical values for k do not provide the expected confidence intervals, k=5 produces error below than 2%. However, k cannot be bounded when 0% error is desired.

Finally, Fig. 8b illustrates the bitrate of an OD pair along the day mixing different metro-flow traffic profiles, as well as the predictions based on the proposed metro-flow model aggregation. Note that min and max models have been obtained for k=5. As expected, because of traffic aggregation, the variability of OD traffic is much smaller than that of metro-flows.

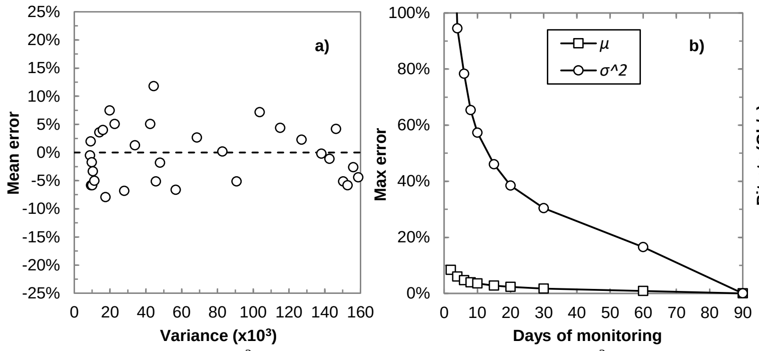


Fig. 6. Mean error of σ^2 estimation (a) and max. error of μ and σ^2 vs. days of monitoring (b).

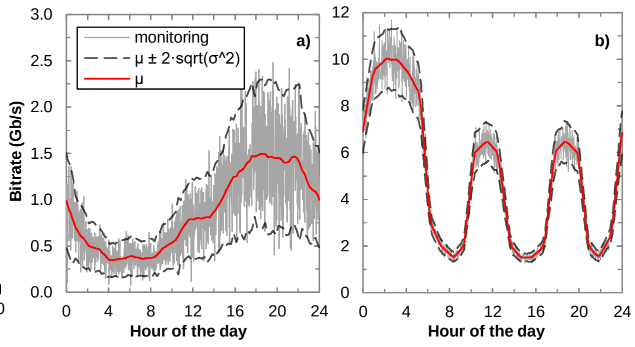


Fig. 7. Prediction of min/max/avg for *Users* (a) and *DC* (b) traffic profiles.

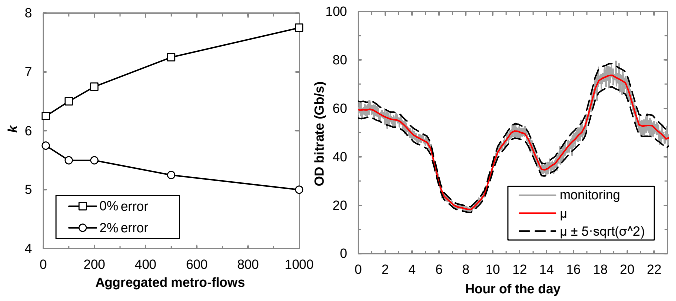


Fig. 8. (a) Value of k (eq. (8)) vs. number of aggregated metro-flows and (b) Prediction of min/max/avg OD bitrate during one day.

C. *VNT reconfiguration performance*

For evaluation purposes, we developed an event driven network simulator and considered a 14-node core VNT interconnecting 7 metro networks using dual-homing, where 100 Gb/s lightpaths support vlinks in the metro areas. A total of 1,400 metro-flows following the *Users* and *DC* profiles were injected into the core VNT and monitored for $|Y|$=3 months to estimate core-based OD models. The same estimation time was used for Metro-flow traffic models.

Once models have been estimated, two metro-flow rerouting actions are triggered daily in all metro areas, being the core controller notified for such changes. The first rerouting action takes place at 7 am and it splits metro-flows leaving the metro area to use the two egress routers to avoid congestion. Two rerouting schemes are considered: *i*) *randomized* and *ii*) *per type of service*. When the *randomized* rerouting scheme is used, metro-flows are evenly split to use the two egress routers, whereas in the *per type of service* scheme only the flows of randomly selected services are rerouted towards the second router. The second rerouting action takes place at 8 pm and groups all metro-flows to use one single egress router.

Under these scenarios, we evaluated two different approaches to reconfigure the core VNT. The first approach, named as *threshold-based* uses a fully meshed VNT and increases or decreases vlinks' capacities according a capacity usage threshold; this approach is followed when the core controller is not able to rebuild obsolete OD traffic models under frequent metro-flow rerouting. In the second approach, the VENTURE algorithm [5] runs periodically, e.g., every hour, to optimize the VNT using OD traffic predictions based on the proposed metro-flow model aggregation updates.

Fig. 9 shows the maximum number of 100Gb/s transponders needed to convey core traffic under the randomized (a) and the per type of service (b) rerouting schemes, for a range of increasing traffic loads. Note that since the number of transponders is not limited in the nodes, both reconfiguration approaches offer zero blocking probability. It can be observed that VENTURE is able to adapt the VNT using less transponders than that of the threshold-based approach in the studied range of loads, producing savings up to 30% under the randomized rerouting scheme and up to 40% when per type of service rerouting is considered.

D. *Experimental assessment*

Experiments to assess the proposed architecture have been carried out in a distributed test-bed connecting CNIT (Pisa, Italy) and UPC (Barcelona, Spain) premises. A core controller with segment-routing capabilities [24] was located at CNIT, whereas the Flow Controller and two metro controllers were located at UPC. The core and metro controllers were extended with an HTTP REST API interface to exchange JSON-encoded messages with the Flow Controller. The SDN controller used in the metro areas is based on RyuSDN and uses OpenFlow to configure the network nodes. Finally, a set of extended nodes [13] implemented on top of OpenVSwitches were deployed using Mininet at UPC premises to allow monitoring traffic. For the experiments, the example in Fig. 1 was reproduced; for the sake of clarity, the captures in Fig. 10 and Fig. 11 are labeled as in Fig. 4.

Fig. 10a lists the REST API messages exchanged between the metro controllers and Flow Controller after a LSP is set-up

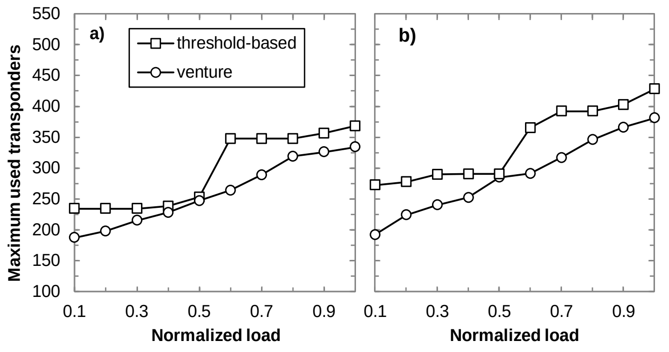
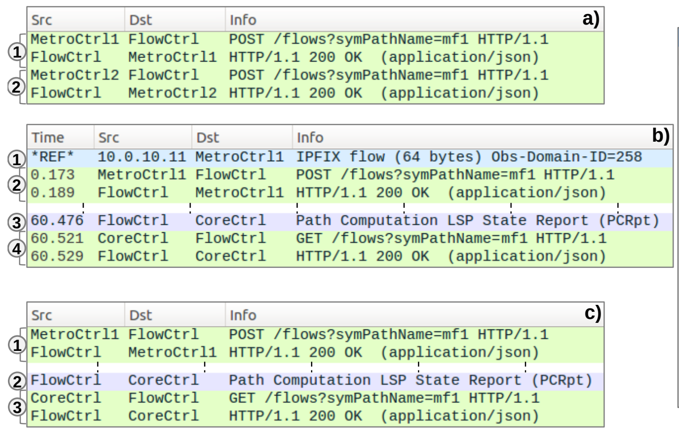


Fig. 9. Maximum used transponders under *randomized* (a) and *per type of service* (b) rerouting schemes.



```
   Src         Dst         Info                                                a)
① MetroCtrl1  FlowCtrl    POST /flows?symPathName=mf1 HTTP/1.1
   FlowCtrl    MetroCtrl1  HTTP/1.1 200 OK  (application/json)
② MetroCtrl2  FlowCtrl    POST /flows?symPathName=mf1 HTTP/1.1
   FlowCtrl    MetroCtrl2  HTTP/1.1 200 OK  (application/json)
```

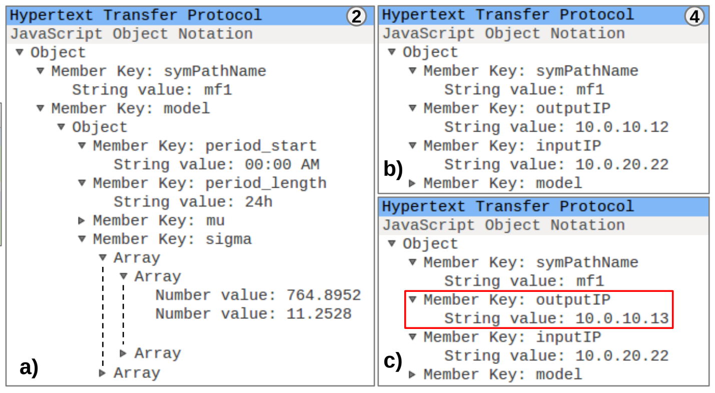
```
   Time    Src          Dst         Info                                           b)
① *REF*   10.0.10.11   MetroCtrl1  IPFIX flow (64 bytes) Obs-Domain-ID=258
② 0.173   MetroCtrl1   FlowCtrl    POST /flows?symPathName=mf1 HTTP/1.1
   0.189   FlowCtrl     MetroCtrl1  HTTP/1.1 200 OK  (application/json)

③ 60.476  FlowCtrl     CoreCtrl    Path Computation LSP State Report (PCRpt)
④ 60.521  CoreCtrl     FlowCtrl    GET /flows?symPathName=mf1 HTTP/1.1
   60.529  FlowCtrl     CoreCtrl    HTTP/1.1 200 OK  (application/json)
```

```
   Src         Dst         Info                                                c)
① MetroCtrl1  FlowCtrl    POST /flows?symPathName=mf1 HTTP/1.1
   FlowCtrl    MetroCtrl1  HTTP/1.1 200 OK  (application/json)

② FlowCtrl    CoreCtrl    Path Computation LSP State Report (PCRpt)
③ CoreCtrl    FlowCtrl    GET /flows?symPathName=mf1 HTTP/1.1
   FlowCtrl    CoreCtrl    HTTP/1.1 200 OK  (application/json)
```

Fig. 10. Messages list for metro-flow set-up (a), metro-flow traffic model update (b) and metro-flow LSP rerouting (c).



```
Hypertext Transfer Protocol                    ②
JavaScript Object Notation
▼ Object
  ▼ Member Key: symPathName
      String value: mf1
  ▼ Member Key: model
    ▼ Object
      ▼ Member Key: period_start
          String value: 00:00 AM
      ▼ Member Key: period_length
          String value: 24h
      ▶ Member Key: mu
      ▼ Member Key: sigma
        ▼ Array
          ▼ Array
              Number value: 764.8952
              Number value: 11.2528

          ▶ Array
a)      ▶ Array
```

```
Hypertext Transfer Protocol                    ④
JavaScript Object Notation
▼ Object
  ▼ Member Key: symPathName
      String value: mf1
  ▼ Member Key: outputIP
      String value: 10.0.10.12
  ▼ Member Key: inputIP
b)    String value: 10.0.20.22
  ▶ Member Key: model
```

```
Hypertext Transfer Protocol
JavaScript Object Notation
▼ Object
  ▼ Member Key: symPathName
      String value: mf1
  ▼ Member Key: outputIP
      String value: 10.0.10.13
  ▼ Member Key: inputIP
c)    String value: 10.0.20.22
  ▶ Member Key: model
```

Fig. 11. Details of traffic model (a), metro-flow (b) and updated metro-flow data (c).

for metro-flow *mf1* across the two metro domains (labeled as in Fig. 4a).

Messages specify the LSP's symbolic path name (*mf1*) and the IP address of the metro border node. Fig. 10b shows an IPFIX flow message (labeled as 1 in Fig. 4b) containing monitoring data from *mf1* that is sent to the source metro controller for traffic model estimation. After collecting enough traffic data, a predictive model is estimated by the metro controller and sent to the Flow Controller in a JSON-encoded REST API message (message 2). The details are shown in Fig. 11a and include the LSP symbolic path name and the data representing the metro-flow predictive model. Next, the Flow Controller issues a PCEP PCReport message to the core controller notifying the new data available for *mf1* (message 3). The PCReport message contains a list of Stateful Request Parameters (SRP) and one LSP object with the LSP's symbolic path name. The core controller then issues a REST-API request with the LSP symbolic name (message 4) to retrieve its data (detailed in Fig. 11b).

Finally, Fig. 10c lists the messages exchanged after rerouting *mf1*. First, the new border output node is sent by the metro controller in a REST API message to the Flow Controller (labeled as 1 in Fig. 4c). Once the Flow Controller updates the metro-flow data, equivalent messages to those for model creation are exchanged with the core controller to allow obtaining updated data for the rerouted LSPs (notice the updated border node in Fig. 11c). The new traffic models for the OD traffic are computed afterward and become available after less than 100 ms from the rerouting notification.

VI. Conclusions

Aggregated metro-flow traffic predictive model is proposed to cope with OD traffic changes in the core as a result of uncoordinated metro-flow rerouting, where the predictive traffic models are used to reconfigure the core VNT. By conveniently aggregating metro-flows after they become rerouted, OD predictive models can be rebuilt fast to keep the predictive capabilities in the core. To obtain quality metro-flow predictive models, an estimation algorithm that allows obtaining models that can be aggregated and evaluated efficiently is presented.

To be able to create the core OD predictive traffic models, a Flow Controller is proposed to allow metro controllers to share metro-flow traffic models with the core controller. Three workflows have been proposed to keep updated metro-flow data in the Flow Controller: triggered either by the estimation of a new predictive traffic model or by a rerouting action. In any case, these actions originated by metro controllers are properly notified to the Flow Controller and eventually to the core controller.

The process of metro-flow modelling was analyzed, concluding that at least 2 months of monitoring data aggregated every 15 minutes are needed to obtain quality predictions for different traffic profiles. The aggregated traffic model was then used as input for a VNT reconfiguration algorithm, which was evaluated against a threshold-based approach used when new models cannot be rebuilt under changing OD traffic, obtaining savings as high as 40% in terms of used transponders.

Finally, the proposed architecture, including the Flow Controller, was experimentally assessed in a distributed test-bed connecting CNIT and UPC premises.

Acknowledgements

The research leading to these results has received funding from the EC through the METRO-HAUL project (G.A. nº 761727), from the Spanish MINECO SYNERGY project (TEC2014-59995-R), and from the Catalan Institution for Research and Advanced Studies (ICREA).

References

[1] F. Morales, M. Ruiz, and L. Velasco, "Core VNT Adaptation Based on the Aggregated Metro-Flow Traffic Model Prediction," in Proc. OFC, 2017.

[2] F. Morales, Ll. Gifre, F. Paolucci, M. Ruiz, F. Cugini, L. Velasco and P. Castoldi, "Experimental Assessment of a Flow Controller for Dynamic Metro-Core Predictive Traffic Models Estimation," in Proc. European Conference on Optical Communication (ECOC), 2017.

[3] L. Velasco, P. Wright, A. Lord, and G. Junyent, "Saving CAPEX by Extending Flexgrid-based Core Optical Networks towards the Edges," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 5, pp. A171-A183, 2013.

[4] 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 Communications Magazine, vol. 53, pp. 159-165, 2015.

[5] F. Morales, M. Ruiz, Ll. Gifre, L. M. Contreras, V. López, and L. Velasco, "Virtual Network Topology Adaptability based on Data Analytics for Traffic Prediction," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 9, pp. A35-A45, 2017.

[6] A. P. Vela, M. Ruiz, L. Velasco, "Distributing Data Analytics for Efficient Multiple Traffic Anomalies Detection," Elsevier Computer Communications, vol. 107, pp. 1-12, 2017.

[7] L. Velasco, A. P. Vela, F. Morales, and M. Ruiz, "Designing, Operating and Re-Optimizing Elastic Optical Networks," IEEE/OSA Journal of Lightwave Technology (JLT), vol. 35, pp. 513-526, 2017.

[8] J. Gama, I. Zliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," in ACM Journal of Computing Surveys, vol. 46, 2014.

[9] M. Malboubi, L. Wang, C. N. Chuah and P. Sharma, "Intelligent SDN based traffic (de)aggregation and measurement paradigm (iSTAMP)," in Proc. of IEEE INFOCOM 2014.

[10] X. Yuan, N. Chen, D, Wang, G. Xie, and D. Zhang, "Traffic prediction models of traffics at application layer in metro area network," Journal of Computer Research and Development, vol. 3, pp. 13, 2009.

[11] B. Chandrasekaran, "Survey of network traffic models," Washington University in St. Louis, vol. 567, 2009.

[12] H Lütkepohl, "Forecasting aggregated time series variables: A survey," Journal of Business Cycle Measurement and Analysis, vol. 2010, pp. 1-26, 2011.

[13] L. Velasco, Ll. Gifre, J.-L. Izquierdo-Zaragoza, F. Paolucci, A. P. Vela, A. Sgambelluri, M. Ruiz, and F. Cugini, "An Architecture to Support Autonomic Slice Networking [Invited]," IEEE/OSA Journal of Lightwave Technology (JLT), 2018 (DOI: 10.1109/JLT.2017.2748233).

[14] W. Wei, *Time series analysis*, Addison-Wesley publications, 1994.

[15] R. V. Hoog, *Introduction to Mathematical Statistics*, Pearson, 2013.

[16] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang and Y. Guan, "Network traffic classification using correlation information", in IEEE Trans. on Parallel and Distributed Systems, vol. 24, pp. 104-117, 2013.

[17] E. W. Grafarend, *Linear and Nonlinear Models: Fixed Effects, Random Effects, and Mixed Models*, Walter de Gruyter, 2006.

[18] B. Claise, B. Trammell, P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol," IETF RFC 7011, 2013.

[19] E. Crabbe, I. Minei, J. Medved, and R. Varga, "PCEP Extensions for Stateful PCE," IETF draft, work in progress, 2016.

[20] M. Ruiz, M. Germán, L. M. Contreras, and L. Velasco "Big Data-backed Video Distribution in the Telecom Cloud," Elsevier Computer Communications, vol. 84, pp. 1-11, 2016.

[21] L. Velasco, A. Asensio, J. Ll. Berral, E. Bonetto, F. Musumeci, V. López, "Elastic Operations in Federated Datacenters for Performance and Cost Optimization," Computer Communications, vol. 50, pp. 142-151, 2014.

[22] D. P. Bertsekas, R. G. Gallager and P. Humblet, *Data networks*, 2nd edition, Prentice-Hall, 1992.

[23] D. Gross, J. F. Shortle, J. M. Thompson and C. M. Harris, *Fundamentals of Queuing Theory*, 4th edition, Wiley, 2008.

[24] A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi, "Experimental Demonstration of Segment Routing," IEEE/OSA Journal of Lightwave Technology (JLT), vol. 34, pp. 205-212, 2016.