A Scalable and Error-Tolerant Solution for Traffic Matrix Assessment in Hybrid IP/SDN networks

Jaime Galán-Jiménez*, Marco Polverini† and Antonio Cianfrani†
*INTIA Research Institute, University of Extremadura, Cáceres, Spain
†DIET Department, University of Rome ”Sapienza”, Rome, Italy
Email: *jaime@unex.es †name.surname@uniroma1.it

***Abstract*—The advent of the Software Defined Networking (SDN) paradigm represents a great opportunity for the definition of new network management solutions. In this work, we focus on the definition and implementation of a novel technique to solve the Traffic Matrix Assessment (TMA) problem from the perspective of an Internet Service Provider. Since the migration from legacy IP networks to fully-deployed SDN ones needs to be incremental due to budget and technical constraints, this paper proposes a mixed measurement and estimation scalable solution for hybrid IP/SDN networks to accurately solve the TMA problem by exploiting the availability of flow rule counters in SDN switches. The performance evaluation shows that our error-tolerant solution is able to assess the TM with a negligible estimation error by only measuring a small percentage of traffic flows, overcoming other state-of-the-art algorithms proposed in the literature. Moreover, the performance analysis of the proposed implementation using the OpenDaylight controller over an emulated network environment, shows that a trade-off between the quality of the assessed TM and its impact on the network in terms of control messages to be sent can be found by properly tuning the number of measured flows.**

***Index Terms*—Traffic Matrix Assessment, Software-Defined Networks, Hybrid IP/SDN, OpenDaylight.**

I. Introduction

THE centralization of the control plane functions achieved by the introduction of the Software Defined Networking (SDN) paradigm, led to the definition of novel Traffic Engineering (TE) solutions able to improve the Quality of Service (QoS) provided to end users. In this direction, a crucial point for the effectiveness of TE strategies is the assessment of the traffic relationships in the network, i.e., the so called Traffic Matrix (TM).

The Ingress-Egress (IE) TM is a data structure reporting the intensity of the traffic flows entering the network from an Ingress node and leaving it through an Egress node. The Traffic Matrix Assessment (TMA) problem is a well investigated, but still open, research topic [1]. The advent of SDN allows the definition of new traffic measurement techniques, thanks to the availability of specific byte counters associated to the entries of flow tables. As an example, in [2] the authors propose an OpenFlow based framework to assess the TM of a network, while in [3] an algorithm to aggregate/de-aggregate flow table entries for measuring specific targeted flows is presented.

One of the main issues in the process of measuring the TM using SDN rules counters is the limited size, in terms of number of installed rules, of the Ternary Content Addressable Memories (TCAMs) used to build the flow tables of SDN switches [4], [5]. Taking this constraint into account, authors of [6] define the *flow spread* parameter as a quantitative indicator of the most attracting flows to be measured. The idea is therefore to limit the number of flows to measure and determine the rest of the TM using a specific estimation algorithm.

However, among the solutions proposed in the literature to estimate a TM, two open issues have to be considered. As first, the measurement-based approaches for SDN assume that a specific IE flow is measurable by means of a single flow rule. Actually, inserting a rule able to match the Ingress node and the Egress node of an incoming packet is not possible, since this information is not present in the packet header; the only available option is to search for a match in the Source and Destination IP fields but, in this way, a specific Origin Destination (OD) flow is measured. Moreover, the presence of measurements errors, mainly due to synchronization issues during the measurements collection phase, is not taken into account and can lead to the not applicability of existing estimation algorithms, such as the popular Tomogravity [7] one.

In this work, we propose a mixed measurement-estimation algorithm able to define IE flow measurement rules using a scalable rule splitting method, and to overcome potential measurement errors using the Fanout estimation algorithm [8]. Our solution is also compatible with a hybrid IP/SDN network, i.e., a network where legacy IP routers and novel SDN switches coexist.

Finally, there is a lack of real implementations to validate the effectiveness of existing solutions, as well as to evaluate their impact on the network performance. For this reason, we also provide an OpenFlow-based implementation for the OpenDaylight controller [9]; we highlight the effectiveness of our solution in the TMA and the limited impact on the network functioning in terms of execution time and signalling traffic.

Therefore, the main contributions of the present paper can be summarized as follows:

- the proposal of a mixed measurement/estimation-based algorithm, namely CofFE-FAN, to assess the TM of a hybrid IP/SDN network;
- the definition of a scalable method to obtain measurements related to IE traffic flows;
- the implementation of the CofFE-FAN algorithm for the OpenDaylight controller;

- the definition of an experimental methodology to evaluate the performance of our CofFE-FAN implementation on an emulated network environment.

The rest of the paper is organized as follows: Sec. II overviews previous works on TM estimation and on the interrelation of that problem with SDN environments. The hybrid IP/SDN TMA scenario considered in this work is described in Sec. III. Sec. IV introduces the idea of rule splitting during the process of TMA, which is exploited by the CofFE-FAN algorithm described in Sec. V, a mixed measurement and estimation-based technique proposed to solve the TMA problem. In Section VI, an experimental methodology to implement CofFE-FAN on an real SDN network is provided, whilst in Section VII a performance evaluation is carried out to show the effectiveness of the proposed solution. Finally, some conclusions are drawn in Section VIII.

II. Related Work

In the research area of traffic measurement and monitoring, SDN represents an opportunity for the definition of new lightweight solutions. The availability of a byte counter associated to each flow rule into the flow tables of the SDN switches, represents the novel aspect with respect to classical IP devices. Then, differently from classical measurement and monitoring traffic tools, such as Netflow [10] and Sflow [11], where extra operations (i.e., packet classification) with respect to the normal data plane processing and dedicated hardware are needed [12], traffic statistics can be collected without increasing the computation load of switches by exploiting the use of SDN counters.

The basic approach of SDN-based traffic measurement tools relies i) on the installation of measurement-related rules, able to measure selected traffic flows, and ii) on the collection of rules statistics at the SDN controller.

Anyway, the definition of such techniques requires to overcome two main limitations. The flow tables of the SDN switches are realized by means of TCAMs, which are known to have limited size [4], [5]. Moreover, the continuous polling performed by the SDN controller to collect traffic statistics on SDN switches, increases the signalling overhead and generates measurement inaccuracy due to the latency in the gathering phase [13], [14].

To cope with the two aforementioned problems, research efforts have been spent in proposing mixed TMA solutions [2], [6], [15]: the idea is to measure only a subset of flows, and then assessing the remaining ones by means of an estimation algorithm. In this way, it is possible to both reduce the memory space required to install new monitoring rules and, as a consequence, the duration and the signalling traffic of the gathering phase. In this direction, it is crucial to have a criterion to identify the most important flows to measure.

In [6], authors propose the *flow spread* parameter as a way to drive the decision process: it represents the variability range of a given traffic flow, calculated by solving a min/max problem with respect to the size of the target flow, being constrained by the routing and the links loads.

In the Largest Flow First algorithm (LFF), proposed in [15], the most important flows to measure are the ones with the highest intensity. Since this information is not known in advance, LFF defines a method to roughly estimate what are the most loaded flows and then, to install monitoring rules so that to measure them.

In [2], Tootoonchian *et al.* present OpenTM, a flow monitoring tool developed for OpenFlow networks. OpenTM uses built-in features of OpenFlow switches to directly and accurately measure the TM with a low overhead. Additionally, OpenTM uses the routing information learned from the OpenFlow controller to intelligently choose the switches from which to obtain flow statistics, thus reducing the load on switching elements.

iSTAMP [16], in turn, proposes an intelligent sampling algorithm to select the most informative traffic flows, using information gathered throughout the measurement process. More precisely, the flow table entries of the SDN switches are dynamically partitioned into two parts: in the first part, flows are aggregated so that to improve the accuracy of the traffic estimation process, while the second portion is dedicated to measure the most rewarding flows.

In [17], authors propose an inference framework which utilizes Kalman filtering to create an accurate and timely TM. In order to reduce the time needed to collect all the statistics from the flow tables, a switch selection strategy is defined. The aim is to minimize the entropy of the estimation process.

Elephant flow detection and TMA in a Data Center network are the main targets of the algorithms described in [18]. The proposed approaches combine the direct measurements offered by SDN and inference techniques based on network tomography to derive a mixed network monitoring scheme. This mixed scheme aims at finding a balance between measurement overhead and accuracy.

Finally, Tian *et al.* propose a framework to solve the TM estimation problem in an SDN-based IP network [19]. In this case, TCAM utilization is improved by guaranteeing that each flow added to a flow table for traffic measurement cannot be derived from other flows already installed in the flow tables of the SDN nodes.

Even though a theoretical evidence of the efficiency of SDN-based measurement solutions can be proven, three important aspects are still not covered when tackling the TMA problem: i) it is assumed that an IE flow can be measured by installing only one flow rule, which is an unrealistic assumption; ii) potential errors collected during the measurement phase are not considered; iii) no real implementations are provided to quantify the effective impact of the proposed solutions on the network functioning. For this reason, in this paper we face these three aspects by proposing the mixed measurement-estimation CofFE-FAN algorithm, and an OpenFlow-based implementation using the well-known OpenDaylight controller.

III. The Hybrid IP/SDN TMA Scenario

As previously introduced, the main goal of this work is to accurately assess the IE TM of an Internet Service Provider (ISP) network. Since the migration of traditional IP networks toward SDN-capable ones is devised to be incremental due to a

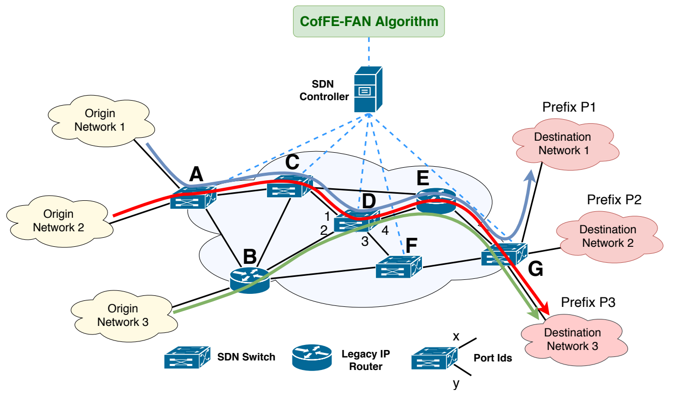


Figure 1: Hybrid IP/SDN TMA system description.

costly investment and the adaptation to a new technology [20], we consider the TMA problem in a hybrid IP/SDN scenario composed of a set of legacy IP routers and a set of upgraded SDN switches [21].

A. The hybrid IP/SDN network scenario

Fig. 1 depicts a hybrid IP/SDN network composed of 2 IP routers and 5 SDN switches. The border nodes, IP routers and/or SDN switches, are able to reach a set of external networks determined by the execution of the Border Gateway Protocol (BGP). In order to set the interior routing, IP routers execute an Interior Gateway Protocol (IGP) to find network paths and fulfil the routing tables, whereas the forwarding tables of SDN switches are directly configured by the SDN controller according to specific policies defined by an algorithm installed on top of it. An interaction between the SDN controller managing the internal domain and the BGP is required to notify the reachability of external networks.

Concerning the definition of traffic flows, we aim to distinguish between OD flows and IE flows. A traffic flow originated by an origin network and destined to destination network is referred to as OD flow. The aggregation of all the OD flows entering the ISP network from the same ingress node and exiting it through the same egress node is referred to as IE flow. Considering Fig. 1, two IE flows are present: the one from node A to node G, composed by two OD flows (i.e., the flow from Origin Network 1 to Destination Network 1 and the flow from Origin Network 2 to Destination Network 3), and the one from node B to node G, composed by a single OD flow (i.e., flow from Origin Network 3 to Destination Network 3).

A flow rule, to be installed in the flow table of an SDN switch, associates a specific action to incoming packets on the basis of its matching condition. In particular, flow rule r is defined as the pair $r = \{Match, Action\}$, where $Match$ can be constructed considering different packet fields from several layers, and $Action$ translates the forwarding policies adopted by the controller to this type of packet.

Regarding management tasks, the byte counters are available for each interface of the IP routers via Network Configuration Protocol (NETCONF) measurements. These values allow the network administrator to know the amount of traffic traversing each IP router. For the case of SDN link load measurements, the SDN controller is able to request information to a particular SDN switch related to the volume of traffic handled by its interfaces.

An important feature of SDN devices is that specific traffic counters, named rule counters, are associated to the flow rules installed in the flow tables of the SDN switches: each of these counters measures the amount of traffic having a matching with the related flow entry. As in the case of link loads, this information can also be retrieved by the SDN controller by means of OpenFlow messages.

B. Model of the TMA problem

Let $\mathcal{G}(\mathcal{N}, \mathcal{L})$ be the graph representing the network of an ISP, where $\mathcal{N}$ is the set of N nodes and $\mathcal{L}$ is the set of L directed links. The set of nodes is divided into IP routers ($\mathcal{N}_{\text{IP}}$) and SDN switches ($\mathcal{N}_{\text{SDN}}$). The external networks reachable by means of the node i are included in the set $\mathcal{E}_i$. If $\mathcal{E}_i \neq \emptyset$ then i is a border node, otherwise it is a transit router. The composition of these sets is determined by the execution of the BGP protocol.

Considering the i-th pair of nodes (s_i, d_i) of the set $\mathcal{N} \times \mathcal{N}$, the quantity x_i reports the intensity of the IE traffic demand entering the network at node s_i and leaving it through the node d_i. The intensities of all IE traffic flows are stored in the vector $\mathbf{x}$, which represents the IE TM.

The path followed by the i-th IE traffic demand is described by the column vector $\mathbf{r}_i$, of length L, where each element $\mathbf{r}_i(l)$ reports the fraction of traffic demand i that is routed over the link l. The routing matrix R is the collection of the vectors $\mathbf{r}_i$ for all possible couple of nodes in $\mathcal{N} \times \mathcal{N}$. The classical TMA problem is then described by the following equation:

$$\mathbf{y} = R \cdot \mathbf{x} \tag{1}$$

where $\mathbf{y}$ is the so called link count vector, in which the element $\mathbf{y}_l$ represents the amount of traffic carried by the link l.

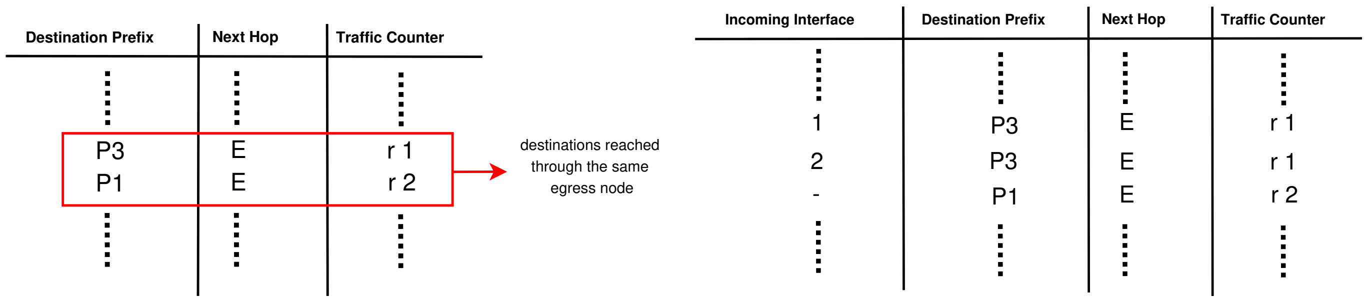
Generally, both $\mathbf{y}$ and R are known in advance, while $\mathbf{x}$ has to be determined. Unfortunately, the linear system reported in Eq. (1) is highly under-determined and then, it admits infinite solutions. The TMA problem consists in selecting the most suitable vector $\mathbf{x}$ that is both a solution of Eq. (1) and as close as possible to the actual TM.

IV. The Rule Splitting Idea to Assess the TM

Starting from the TMA problem reported in Eq. (1), our aim is to insert the information provided by the SDN rule counters so that to reduce the under-determination of the linear system. To do that, we perform two different operations:

- the rule counters aggregation;
- the rule counters splitting.

The *rule counters aggregation* operation is needed since the rule counters are able to measure OD flows, while Eq. (1) requires information about IE flows. In this way, the *rule counters aggregation* operation allows to aggregate the rule counters on the basis of the Egress nodes the flows are directed



Destination Prefix	Next Hop	Traffic Counter
⋮	⋮	⋮
P3	E	r 1
P1	E	r 2
⋮	⋮	⋮

destinations reached through the same egress node

(a)

Incoming Interface	Destination Prefix	Next Hop	Traffic Counter
⋮	⋮	⋮	⋮
1	P3	E	r 1
2	P3	E	r 1
-	P1	E	r 2
⋮	⋮	⋮	⋮

(b)

Figure 2: Example of the SDN forwarding table of node "D" before (a) and after (b) the rule splitting operation.

to. Further details about this operation will be provided in the next section.

The *rule counters splitting* operation represents one of the main contribution of the paper: it allows to extract new information from rule counters performing simple modifications to the SDN flow tables, i.e., inserting new rules extracted by splitting the existing ones. The key idea is that the new rules will provide traffic information with a higher granularity with respect to link load information; at the same time, with respect to existing solutions, our approach is highly scalable since the granularity level of measurements is lower than the ones provided by OD flow measurements, i.e., the insertion of rows matching both IP Source and IP Destination addresses.

The core operation proposed is the de-aggregation of an existing flow rule on the basis of the incoming interfaces: this procedure is referred to as *father-child* splitting. Given an existing rule, it is possible to increase the granularity level of its counter by splitting the rule into two (or more) rules, where the "Incoming Interface" is used as a new matching field. In this way two (or more) counters will be available, providing different traffic related information: given two counters obtained by a *father-child* splitting of an original flow rule, the IE flows measured by a counter (*child*) will be different than the ones measured by a different one (different *child*).

To better explain the *father-child* splitting procedure let us consider the network scenario of Fig. 1 and let us focus on node D, whose (simplified) flow table is reported in Fig. 2a. In the flow table (Fig. 2a), where the matching field is only the IP Destination address, two different rules are reported: one for the traffic directed to Destination Network 3 and one for traffic directed to Destination Network 1. Let us focus on the first rule: it allows to measure the aggregated traffic information related to the green and the red OD flows of Fig. 1. These two OD flows belong to different IE flows: i) the green one is a portion of the IE flow between nodes A and G, while ii) the blue one is a sub flow of the IE traffic between nodes B and G. To increase the IE granularity level of the counter information, it is needed to split the (*father*) rule into two different rules (*children*): one for the green flow and the other one for the blue one.

Looking at Fig. 1, it is clear that the two OD flows have different incoming interfaces for the switch D. In other words, it is possible to de-aggregate the contribution of these two flows by installing two new flow rules in the switch D: each one of them will have exactly the same fields of the *father* rule, with an extra matching field related to the incoming interface. In Fig. 2b the new table after the splitting procedure is reported. Clearly, these *children* rules must have a higher priority with respect to the *father* one. It is worth to mention that, generally, this modification does not lead to the measurement of a single IE flow, but simply allows to perform measurements with a thinner granularity with respect to the initial rules.

V. CoFFE-FAN Algorithm for TM assessment

In this section we present the CofFE-FAN algorithm, a mixed measurement and estimation based technique, whose aim is the assessment of the IE TM in a hybrid IP/SDN network. CofFE-FAN algorithm consists in the execution of three main functional blocks: i) the flow rule counters aggregation, ii) the de-aggregation selection, and iii) the estimation phase. Next, we describe the details of each functional block.

A. *Flow Rule Counters Aggregation*

The main goal of this functional block is to aggregate, for each SDN switch, the traffic related information available in the flow table, according to the egress points. We point out that this function does not modifies the rules installed in the flow tables, nor installs new rules. It simply processes the data stored in the flow tables of the SDN switches to extract an aggregated measure.

In order to clarify this aspect, let us refer to the situation depicted in Fig. 2a, reporting a portion of the flow table of the switch D of Fig. 1. The first rule allows to measure the summation of the green and the red OD flows, while the second one is intended to measure the blue OD flow. All these flow rules measure traffic flows which leave the network from the same egress node. Since the final objective is the assessment of the IE TM, we need to aggregate the measurements performed by different flow rules that are related to destinations reached by means of the same egress node. Specifically, these rules are all the ones such that the network destination prefix is contained in the same set $\mathcal{E}_i$.

We refer with the symbol $\mathcal{F}_i$ to the set of flow rules installed at node $i \in \mathcal{N}_{\text{SDN}}$. Each rule is composed by the following

Algorithm 1 *flow_rule_counters_aggregation* function

```
Require: $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, $\{\mathcal{F}_i\}_{\forall i \in \mathcal{N}_{\text{SDN}}}$, $\{\mathcal{E}_i\}_{\forall i \in \mathcal{N}}$, $R$
1: define $R^{\text{SDN}} = \emptyset$ and $\mathbf{y}^{\text{SDN}} = \emptyset$
2: for all $i \in \mathcal{N}_{\text{SDN}}$ do
3:     for all $n \in \mathcal{N}$ do
4:         tmp =$aggregate\_counters(\mathcal{F}_i, \mathcal{E}_n)$
5:         $\mathbf{y}^{\text{SDN}}.append(\text{tmp})$
6:         $\mathbf{r}^{\text{SDN}}$ =$select\_flows(R, n)$
7:         $R^{\text{SDN}}.append(\mathbf{r}^{\text{SDN}})$
8:     end for
9: end for
```

three information[1]: i) the destination network prefix, ii) the next hop node, and iii) the flow rule traffic counter. The pseudo code of the *flow_rule_counters_aggregation* function is reported in Algorithm 1.

The *flow_rule_counters_aggregation* function takes as input the network graph, the flow tables of the SDN switches and the set of destinations reachable by means of each network node. It produces as output, a matrix R^{SDN} related to the IE flows passing through the SDN switches and the vector $\mathbf{y}^{\text{SDN}}$ of the traffic measurements achieved by the initial installed flow rules. In line 1, the matrix R^{SDN} and the vector $\mathbf{y}^{\text{SDN}}$ are defined as empty. Then (line 2), at turn, the flow table of each SDN switch i is considered. Egress nodes n are analysed one at time (line 3), and for each of them the function *aggregate_counters* is applied. This function searches the subset of rules installed in the flow table of node i having as destination a network prefix reachable by means of the egress node n. Then, the summation of the traffic counters related to these rules is returned as output. The result of the previous operation is stored in the $\mathbf{y}^{\text{SDN}}$ vector (line 5). Moreover, in line 6, by means of the function *select_flows*, the row vector $\mathbf{r}^{\text{SDN}}$ is computed. Each component of this vector represents the fraction of an IE flow steered through the SDN switch i and leaving the network through egress node n. In line 7, this vector is appended to the matrix R^{SDN}.

The output of the *flow_rule_counters_aggregation* function can be used to extend the system in Eq. (1) as follows:

$$\begin{bmatrix} R \\ R^{\text{SDN}} \end{bmatrix} \cdot \mathbf{x} = \begin{bmatrix} \mathbf{y} \\ \mathbf{y}^{\text{SDN}} \end{bmatrix} \tag{2}$$

In [22] it is shown the benefit achieved by this new information on the resolution of the TMA problem.

B. De-aggregation Selection

Despite *flow_rule_counters_aggregation* function execution allows the reduction of the gap between the number of unknowns and the rank of the matrix describing the TMA problem, this last is still under-determined (this is true in hybrid scenarios, as proven in [22]). Further measurements can be performed by properly defining a set of new flow rules to install on the flow tables of the SDN switches. In CofFE-FAN algorithm, this is done by defining the de-aggregation operation.

A de-aggregation operation consists in the application of the *father-child* splitting strategy introduced in section III on the set of rules installed on a given SDN switch that are related to the same egress node. Specifically, considering an SDN switch i, an egress node n, and a link l entering the node i, then the 3-tuple (i, n, l) refers to the recursively application of the *father-child* splitting on all the flow rules installed at node i, such that the specified destination prefix is reached by means of the egress node n. The *children* rules will have the same fields of the *father* one, plus the specification of link l as input port.

Clearly, the cost of a de-aggregation operation is evident: a set of new rules must be installed in the flow table of an SDN switch. More precisely, the number of new flow rules produced by a de-aggregation operation (i, n, l) is $\mathcal{O}(|\mathcal{E}_n|)$. Since the TCAMs used to build the flow tables are limited in size [4], [5], a strategy to select the most suitable de-aggregation operations needs to be defined. To be convenient, a de-aggregation operation must lead to a consistent reduction of the uncertainty in the TMA problem.

For this reason, we define a score function to decide whether a de-aggregation operation is convenient or not. The score exploits the idea of *flow spread* defined in [6], a parameter that provides an indication of the width of the variability range of the intensity of a flow in the TMA problem. Considering the de-aggregation operation (i, n, l), its score $S_{(i,n,l)}$ is defined as the summation of the *flow spread* of all the IE flows entering the switch i by means of link l and leaving the network through the egress node n.

Once the scores of all possible de-aggregation operations have been calculated, the *flow_rule_de-aggregation* function, whose pseudo code is reported in Algorithm 2, is executed.

The *flow_rule_de-aggregation* function takes as input the routing matrix, the matrix R^{SDN} of the routing involving only the SDN switches, the set of all de-aggregation operations, the available space of the TCAMs of the SDN switches (α_i), the set of external networks reachable through border nodes and the flow tables of the SDN switches. It returns as output: i) the matrix R^* that specifies the relation between each selected de-aggregation operation and the IE traffic flows, and ii) the set $\mathcal{M}$ of selected new measurement rules.

As first (line 1), the data structure that will contain the output is defined as empty. The extended routing matrix R^{ext}, given by the union of the matrices R and R^{SDN}, is defined at line 2. Then, SDN switches are considered one at time (line 3). For each of them, an ordered list of de-aggregation operations is created by using the *sort* function. The criterion is to order the items of the set $\{(i, n, l)\}_i$ from highest score to lowest. After that, the list is inspected (line 5). If the TCAM of the considered SDN switch still has room to host new flow rules (line 6), then the algorithm analyses the impact of the current de-aggregation operation on the extended system. To do that, as first we need to define the equation that can be written by performing the operation under test. This is done (line 7) by means of the *select_flows* function, which returns a row

[1]Flow rules are assumed to have this structure due to the considered scenario, i.e., the one of a hybrid IP/SDN network.

Algorithm 2 *flow_rule_de-aggregation* function

```
Require: $\{(i,n,l)\}_{\forall i \in \mathcal{N}_{\text{SDN}}}$, $\{\alpha_i\}_{\forall i \in \mathcal{N}_{\text{SDN}}}$, $\{\mathcal{E}_n\}_{\forall n \in \mathcal{N}}$, $R$, $R^{\text{SDN}}$, $\{\mathcal{F}_i\}_{\forall i \in \mathcal{N}_{\text{SDN}}}$
 1: define $R^* = \emptyset$, $\mathcal{M} = \emptyset$
 2: $R^{\text{ext}} = \begin{bmatrix} R \\ R^{\text{SDN}} \end{bmatrix}$
 3: for all $i \in \mathcal{N}_{\text{SDN}}$ do
 4:     list $=$sort$(\{(i,n,l)\}_i, \{S_{(i,n,l)}\}_i)$
 5:     for all $(i,n,l) \in$ list do
 6:         if $\alpha_i \geq |\mathcal{E}_n|$ then
 7:             $\mathbf{r}^* =$select_flows$(R,n,l)$
 8:             if rank$(\begin{bmatrix} R^{\text{ext}} \\ \mathbf{r}^* \end{bmatrix}) >$ rank$(R^{\text{ext}})$ then
 9:                 $R^{\text{ext}}$.append$(\mathbf{r}^*)$
10:                 $R^*$.append$(\mathbf{r}^*)$
11:                 new_rules $=$de_aggregate$(F_i, (i,n,l), E_n)$
12:                 $\mathcal{M}$.insert(new_rules)
13:                 $\alpha_i - = |$new_rules$|$
14:             end if
15:         end if
16:     end for
17: end for
```

vector whose j-th element is the percentage of the IE flow j that enters the node i by means of the link l, and leaves the network through the egress node n. Next (line 8), the impact of the de-aggregation operation under test on the extended system is evaluated. If the insertion of the row vector $\mathbf{r}^*$ on the matrix R^{ext} produces an increase of the rank, then the de-aggregation (i,n,l) is considered and the algorithm status is updated (lines 9-13). In particular, the *de_aggregate* function (line 11), finds the *fathers* rules, i.e., all the flow rules in the TCAM of the considered switch i that are related to destinations reached through the egress node n. Then, children rules are generated by adding an additional field to the *father* rules specifying the link l as input port, and a higher level of priority. In case the current de-aggregation operation does not allow to reduce the uncertainty of the TMA problem, the algorithm starts analysing the next item in the list.

Once the *flow_rule_de-aggregation* function is performed, the controller starts to install the new measurement rules contained in $\mathcal{M}$.

C. TM Estimation

Once the new flow rules have been installed, it is possible to obtain the related measurements. Let us refer to this set of new measurements as $\mathbf{y}^{*2}$. The TMA problem can now be formalized as follows:

$$\begin{bmatrix} R \\ R^{\text{SDN}} \\ R^* \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{y} \\ \mathbf{y}^{\text{SDN}} \\ \mathbf{y}^* \end{bmatrix} \tag{3}$$

Depending on the availability of free space in the TCAMs of the SDN switches, the number of new flow rules that the ISP operator wants to install, and other factors, the linear system of Eq. (3) can or cannot have full rank. In case it has, then the TM can be easily calculated by inverting Eq. (3); otherwise, an estimation algorithm is needed to complete the assessment procedure.

In the CofFE-FAN algorithm we use the Fanout estimator [8]. This choice is motivated by the following reasons: i) it has low computational complexity; ii) it allows to enforce the estimation by using a temporal sequence of measurements; and iii) it produces an estimation also by using measurements affected by errors.

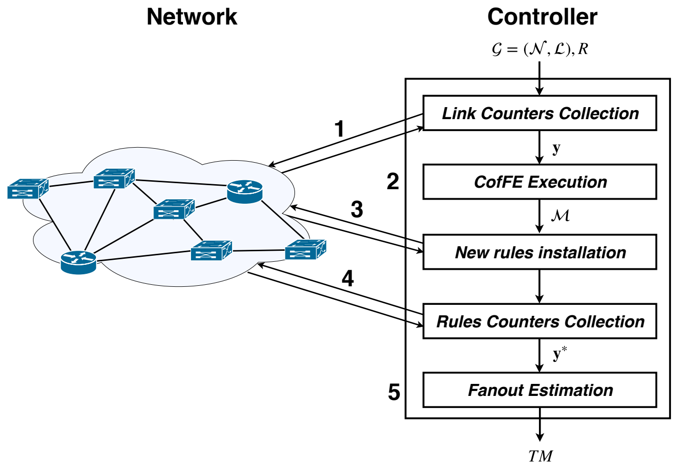


Figure 3: Overview of the *CofFE-bundle* block diagram.

VI. CofFE-FAN on OpenDaylight

In this section, an experimental methodology to implement the CofFE-FAN algorithm described in Section V on an SDN network is provided. More in detail, the different phases of the process, from the collection of links and rules information on SDN switches to the computation of the estimated TM are described. The considered SDN controller is OpenDaylight [9] with OpenFlow version 1.5.0 [23]. A java-based bundle, namely *CofFE-bundle*, is implemented to be therefore integrated into the OpenDaylight controller. The main procedures of *CofFE-bundle* are reported in Fig. 3:

1) the *Link Counters Collection* procedure takes the network topology $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ as input to assess the link count vector $\boldsymbol{y}$ after measuring the link loads for each link $l \in \mathcal{L}$.
2) the *CofFE Execution* procedure finds a subset of new flow rules $\mathcal{M}$ to be installed in order to get new traffic measurements.
3) the *Rules Counters Collection* procedure assesses the byte count vector $\boldsymbol{y}^*$ after measuring the byte counters of the rules reported by the *CofFE Execution* procedure.
4) the *Fanout Estimation* procedure finally executes the Fanout estimator [8].

In the following subsections, each of the aforementioned procedures is described in detail.

Link Counters Collection: An important feature in order to accurately estimate a TM is to have a reliable and effective

[2]note that the i-th element of this vector is given by the summation of the rule counters associated to flow rules related to the i-th selected de-aggregation operation.

monitoring tool [24] providing updated information about the network state.

For this purpose, OpenFlow protocol provides a good support reporting the required information through *OFPT_STATS_REQUEST* and *OFPT_STATS_REPLY* messages. The *OFPT_STATS_REQUEST* message is used by the controller to request statistics for a specific switch. The corresponding switch creates an *OFPT_STATS_REPLY* message reporting values related to different types of statistics, such as *OFPST_PORT* (port statistics), *OFPST_FLOW* (flows statistics), or *OFPST_TABLE* (table statistics), among others.

The particular function used by the OpenDaylight controller to request the link load (in terms of number of bytes) on $l \in \mathcal{L}$ is *l.getTransmitByteCount()*, which returns the total transmitted byte counter for the specified port. In order to obtain the load on links connecting IP routers, we rely on the information provided by the NETCONF protocol.

When the *Link Counters Collection* process ends, the link count vector $\boldsymbol{y}$ is ready to be passed as input for the next procedure, i.e., the execution of the CofFE-FAN algorithm.

CofFE Execution: Taking as input the link count vector y reported by the *Link Counters Collection* procedure and the routing matrix R, CofFE-FAN returns a set of flow rules to be installed on a subset of nodes, as described in Section V. Moreover, CofFE-FAN is also responsible for notifying the controller to send the corresponding *OFPT_FLOW_MOD* messages to the nodes.

Rules Counters Collection: In this step, the controller sends a message of type *OFPT_STATS_REQUEST* to the nodes where the new flow rules have been installed, and analyses the information related to flow statistics (*OFPST_FLOW*) from the *OFPT_STATS_REPLY* received messages. The function provided by OpenDaylight to get the number of bytes that matched a particular flow entry r installed on a node $n \in \mathcal{N}_{SDN}$ is *n.getByteCount(r)*. Statistic values for flow counters are assessed after Δ_t seconds, in the same way as for link counters collection. Next, the measurements provided by flow rules related to the same de-aggregation operation need to be aggregated, in order to obtain the byte count vector $\boldsymbol{y}^*$.

Fanout Estimation: Finally, the Fanout estimator [8] is applied to assess the resulting TM, which is reported as output.

VII. Performance Evaluation

In this section, a performance evaluation is carried out to show the effectiveness of the CofFE-FAN algorithm. As first, we compare the CofFE-FAN algorithm performance with the ones achieved by two TMA solutions available in the literature: the Flow Spread Based Algorithm (FSBA) described in [6], and the Largest Flow First (LFF) algorithm presented in [15]. Next, we consider an emulated environment in order to show the performance of the proposed implementation of CofFE-FAN on the OpenDaylight controller. For this purpose, several tests have been conducted on two emulated networks by means of the Mininet emulation tool [25]. In particular, two analysis have been performed: i) the first analysis is a comparison among the performance obtained using simulations (i.e., theoretically) with the ones obtained through emulation; and ii) the second analysis aims at showing the set of events that occur when CofFE-FAN algorithm runs to accurately estimate a TM.

A. Dataset for Simulation

For our experiments we selected real network topologies from the SNDLib database [26], where real traffic matrices are also available. We select four different networks: Abilene ($N = 12$, $L = 30$), Nobel ($N = 17$, $L = 52$), Geant ($N = 22$, $L = 72$) and Germany ($N = 50$, $L = 176$).

The following strategy is used to generate a hybrid IP/SDN scenario: i) as first, the percentage of SDN switches in the network is chosen, then ii) the nodes are sorted according to their betweenness centrality [27], and finally iii) the first nodes of the list are selected to be SDN switches (the remaining ones are IP routers). Each SDN switch has a TCAM able to host up to 4000 rules (as suggested in [4]). A number K of external networks is reachable by means of each network node.

The IGP routing is determined by means of the Dijkstra algorithm, i.e., by following a shortest path and destination based rule, while for the external networks a hot potato policy is applied. The routing tables of the IP routers, as well as the flow tables of the SDN switches are fulfilled accordingly. A flow rule specifies the next hop node and the destination network. Initially, each SDN switch has $K \times N$ installed rules.

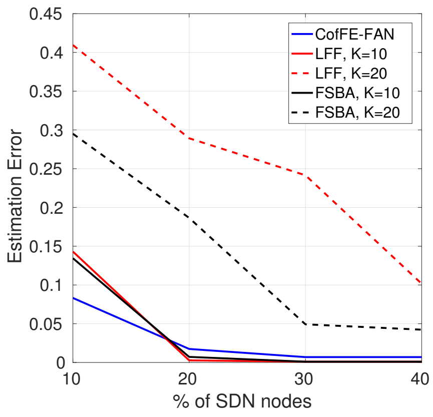
Two existing algorithms, named FSBA and LFF, are used as benchmark for the CofFE-FAN algorithm. Both of them are mixed measurement and estimation based approaches, thought to work in a hybrid IP/SDN environment. Differently from CofFE-FAN algorithm, which exploits the concept of rule de-aggregation, the selected approaches aim at measuring IE traffic flows by installing specific flow rules. While both FSBA and LFF use the Tomogravity model [7] in the estimation phase, they differ in the method used to select the set of IE flows to measure: i) FSBA uses the *flow spread* parameter to drive the selection process, while ii) LFF tries to measure the largest flows.

Finally, we choose the Relative Root Mean Squared Error (RRMSE) [28] as parameter to quantify the quality of the assessed TM.

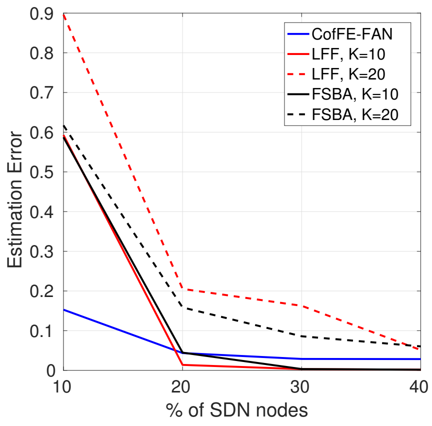
B. Performance Analysis in a Simulated Environment

In the considered hybrid IP/SDN network, the percentage of SDN switches highly affects the performance of the TMA procedure. In fact, as first the number of SDN switches poses an upper limit to the number of flow rules that can be installed for traffic measurement purposes. Moreover, some of the IE flows might not be measurable, due to the fact that they do not traverse any SDN switch.

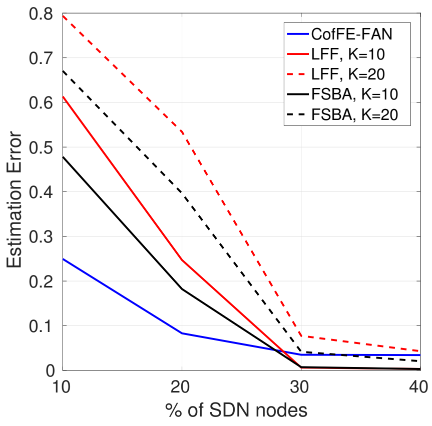
For this reason, the first analysis we propose aims at evaluating the estimation error as a function of the percentage of SDN switches in the considered hybrid network. The results of this study are reported in Fig. 4. As expected, the estimation error obtained by the three considered approaches decreases as the percentage of SDN switches in the network increases. When the percentage of SDN switches in the network is below 20%, CofFE-FAN algorithm gets better results than both FSBA and LFF. There are two main motivations for this situation: i) as



(a) Nobel

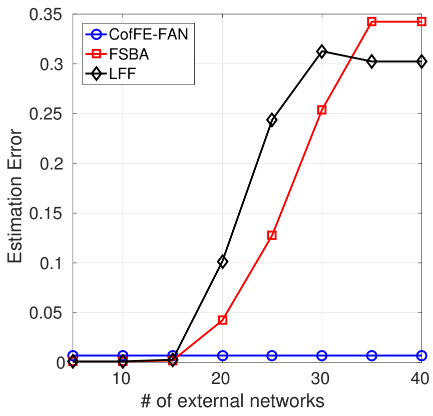


(b) Geant

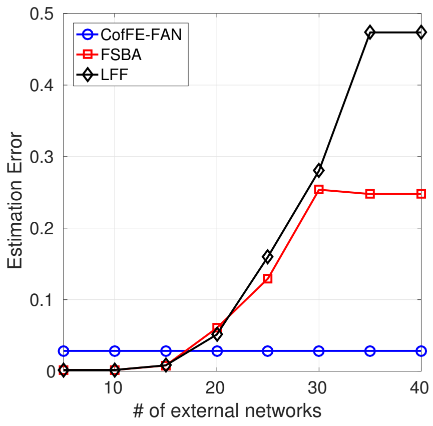


(c) Germany

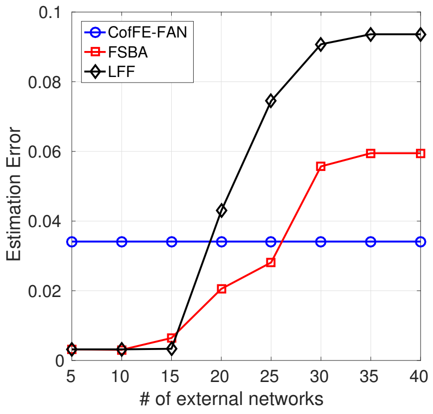
Figure 4: Estimation Error as a function of the percentage of SDN switches.



(a) Nobel

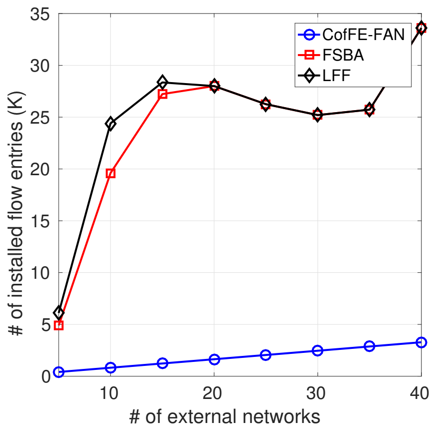


(b) Geant

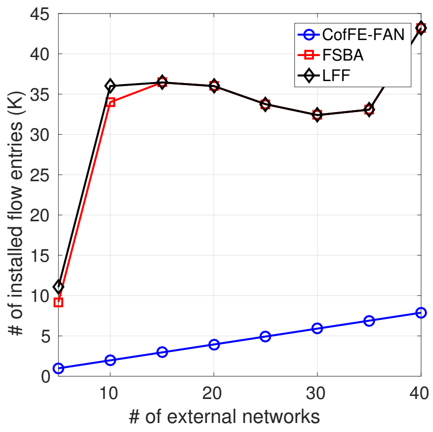


(c) Germany

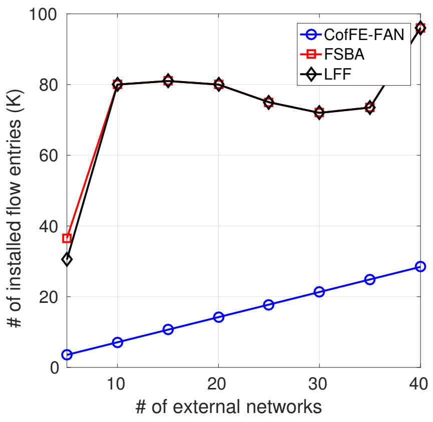
Figure 5: Estimation Error as a function of the number of external networks.



(a) Nobel



(b) Geant



(c) Germany

Figure 6: Number of installed flow rules as a function of the number of external networks.

first, in these conditions, the FSBA and LFF algorithms suffer, since the IE flows that are more appealing with respect to the considered criterion, might be not measurable, due to the fact that they do not find any SDN switch along their path; ii) secondly, the Fanout estimator provides better estimation than the Tomogravity one. Another outcome of the analysis reported in Fig. 4 is that, for $K = 20$, CofFE-FAN algorithm always overcomes the other two algorithms. To explain this behaviour, we point out that the main advantage of the de-aggregation strategy used by CofFE-FAN algorithm is that, the number of installed flow rules due to a de-aggregation operation grows linearly with K. On the contrary, considering FSBA and LFF, the measurement of an IE traffic flow requires the installation of a number of rules that grows with the square of K. This is due to the fact that an IE flow consists of the aggregation of the OD flows between all the source networks reachable by means of the ingress node, and all the destination networks that can be reached through the considered egress node.

In order to better investigate the potential advantage achieved by the use of CofFE-FAN algorithm, we propose an analysis of the estimation error as a function of the K parameter. The results of this study, reported in Fig. 5, refer to the case of a percentage of SDN switches equal to 50%. A first remarkable outcome of this analysis is that the quality of the TM assessed by means of CofFE-FAN algorithm is independent from K. Moreover, the achieved error is generally negligible. This is a further evidence of the high level of scalability reached by the proposed approach. On the contrary, the performances of both FSBA and LFF algorithms, are highly affected by the number of external networks. More precisely, for low values of the K parameter, the two benchmark solutions achieve better results than CofFE-FAN algorithm. Anyway, as soon as the number of external networks overcome a given threshold, the estimation error of the TM produced by FSBA and LFF considerably increases. It is interesting to notice that the threshold value depends on the size of the network. This is due to the fact that, in bigger networks, the overall number of measurement rules that can be installed is higher, and consequently, the scalability issues are experienced for higher values of K.

The previous analysis have highlighted that CofFE-FAN algorithm outperforms the other considered solutions. The lower estimation error is not the only advantage of CofFE-FAN algorithm. In fact, the high efficiency of the proposed rule de-aggregation technique is also reflected in the number of flow rules that have to be installed with the only purpose of measuring traffic. In order to show this further aspect, we report in Fig. 6 the number of flow rules installed by the considered algorithms, as a function of the K parameter. Before commenting the results, we clarify the main benefits of reducing the number of flow rules to install:

- it increases the efficiency in the use of the TCAMs;
- it reduces the time needed to install the rules, making the process of TMA faster;
- it decreases the overhead due to the signalling messages sent by the controller to install the rules and to collect the traffic counters;
- it reduces the time needed to collect the traffic counters, improving the overall execution time, and reducing the problems related to the synchronization of the measurements (one of the main sources of measurement errors).

First of all, looking at Fig. 6 it is evident the huge reduction of the number of measurement rules achieved by CofFE-FAN algorithm with respect to the other considered solutions. Furthermore, considering CofFE-FAN algorithm, it is interesting to notice that the number of installed flow rules increases linearly with respect of to the K parameter. On the opposite, both FSBA and LFF require the installation of a very high number of flow rules. For them, two different regions can be identified: i) for values of K below a threshold, the number of installed flow rules grows with a law that is approximately $2 \times K^2$; ii) when K reaches higher values, then this relation changes. This behaviour is justified by the fact that, in the first region, both the number of measured flows and the number of rules needed to measure a flow increases, while, when the TCAMs of the SDN switches reach the saturation level, then the number of measured flows decreases with K, while the number of rules needed to measure a flow still increases.

C. Dataset for Emulation

Due to scalability issues, for the emulation experiments we consider a subset of the networks used in the simulation case: Abilene and Nobel. The setting of the emulated scenario is realized in the following way:

- an SDN switch is associated to each network node;
- a dedicated host is connected to each node in order to serve as source/destination of traffic;
- traffic flows among each source-destination pair are established using the iPerf tool [29].

We use OpenDaylight to implement the SDN controller [9], *openvswitch* to implement the set of SDN nodes and OpenFlow 1.5.0 [23] for the southbound communication among the controller and the nodes. Tests are performed on a dual-core Intel-based machine (3.1 GHz) with 16 GB of RAM.

Experiments are executed as follows. As first, the SDN controller is started and the java-based *CofFE-bundle* is installed on top of it. The network topology is loaded by Mininet, and the routing paths are configured by installing proper rules at the SDN switches by means of specific OpenFlow messages. Without loss of generality, we consider destination based and shortest path routing, computed executing the Dijkstra algorithm. In this way, at each SDN node there are as many entries as the total number of destination hosts. The OpenFlow message sent by the controller to install a new rule on a node is the flow table modification message *OFPT_FLOW_MOD*, with *OFPFC_ADD* command.

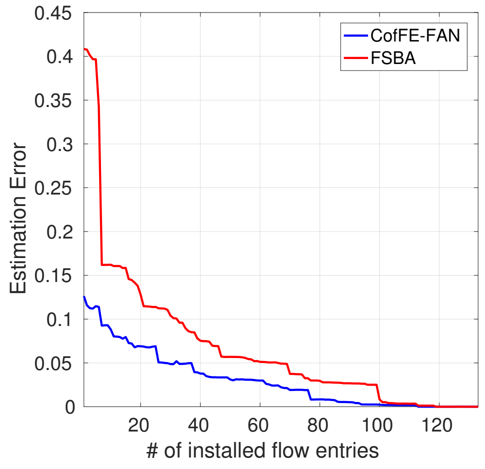
After installing the set of initial flow rules, the traffic demands retrieved from the TM taken as input are sent during a time period of $\Delta_t = 60$ seconds using iPerf [29]. In order to do this, UDP traffic is generated between the hosts connected to SDN switches. Link counters values are then retrieved for all links after a specific time interval, δ_t. In the experiments, we set $\delta_t = 5$ seconds in order to allow the system to be stable after topology load, rules installation and the beginning of traffic delivery. Once the set of new flow rules is obtained (as a result of the execution of CofFE-FAN), the controller sends the corresponding OpenFlow messages to install them in the selected nodes. Finally, after $\Delta_t = 60$ seconds, rules counters are obtained.

In the emulated environment, we need to introduce some modifications on the normal work flow of the considered algorithms (CofFE-FAN and FSBA in this second set of analyses). As first, due to limited hardware resources, only one host is connected to each SDN switch. In this conditions, an initial rule can potentially measure a single IE flow, hiding the benefits of running CofFE-FAN algorithm. Then, in the emulation environment we do not consider the measurements associated to the initial rules. In order to compensate this loss of information, we decide to consider a full SDN network. Additionally, since we cannot generate enough OD flows to stress the TCAMs of the SDN switches, we change the way in which the considered algorithms select the measurement rules. More in detail, we provide as input parameter to the algorithms the maximum number of new rules that can be installed (instead of the size of the TCAMs).

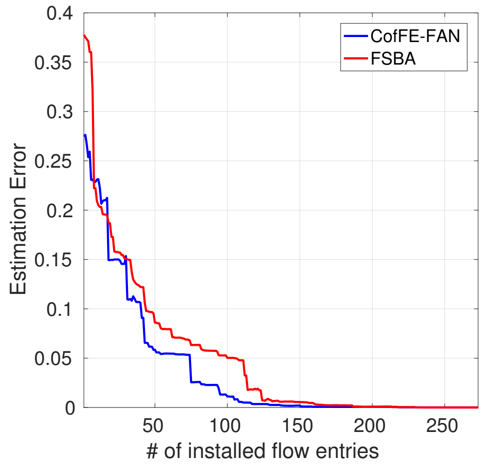
D. *Performance Analysis in a Emulated Environment*

The first analysis we propose aims at verifying whether, under the new changes, CofFE-FAN algorithm still outperforms the FSBA solution. For this reason, we evaluate the estimation error obtained by the two algorithms, as a function of the maximum number of new flow rules that can be installed. The results reported in Fig. 7 are obtained through simulation, but considering the modifications introduced in the behaviour of the two algorithms. As it can be seen, CofFE-FAN algorithm still outperform FSBA.This is mainly due to the use of Fanout estimator, which increases the quality of the estimation with respect to the simple Tomogravity algorithm. This advantage is more evident looking at Fig. 7 when no one new flow rule is installed. In this case, the estimation is carried out by considering only the links loads. As it can be seen, in case of Abilene, the estimation error of CofFE-FAN algorithm is about 0.1, while FSBA obtains a value of 0.4.

The next analysis we propose aims at comparing CofFE-FAN algorithm and FSBA, when real measurements are considered. In fact, differently from the previous comparison, where simulated data are considered, in the results reported in Fig. 8 the measurements are taken from the emulated environment. The main difference is that, these measurements are affected by potential errors due to several factors, such as packet loss, synchronization among measurements, etc. The most important outcome of the analysis reported in Fig. 8 is that, while FSBA can fail in assessing the TM, CofFE-FAN algorithm always produces a result. This is due to the use of the Fanout estimator, that is tolerant to errors and to incoherence between measurements. In particular, until the number of considered measurements is relatively small, both algorithms provide a result, even though it is evident that CofFE-FAN algorithm produces a more accurate estimation. Then, when the number of measurements increases, then Tomogravity fails in producing an output. This is one of the



(a) Abilene



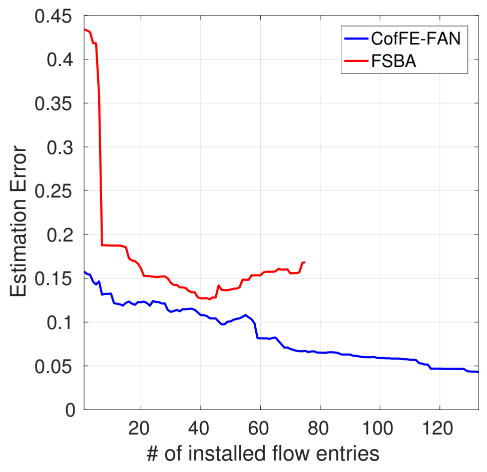
(b) Nobel

Figure 7: Estimation error achieved by CofFE-FAN algorithm and FSBA as a function of the number of installed flow entries.

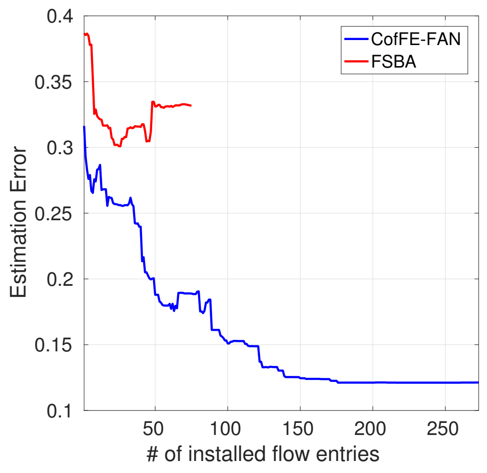
main advantages of CofFE-FAN, i.e., its ability in coping with possible errors in the obtained measurements. This allows to use it in real environments.

In order to quantify the loss of performance experienced by CofFE-FAN algorithm when using measurements affected by errors, in the next analysis we compare the estimation error as a function of the number of installed rules, when taking as input simulated data or measurements obtained in the emulated environment. The results of this study are reported in Fig. 9. As first, we notice that the gap among the results obtained in the simulation environment and in the emulated one grows with the number of installed rules. This is due to the fact that in the emulated environment the measurements are affected by errors, which make the estimator to deviate from the actual TM. Moreover, a further negative effect is that, the final result converges to a higher value of estimation error. Despite this, the overall quality of the estimation is sufficiently high.

After comparing the results obtained by simulations with the ones retrieved from emulations, the purpose of next analysis is to characterize the impact of our proposal on the network performance in terms of: i) time required by the different tasks

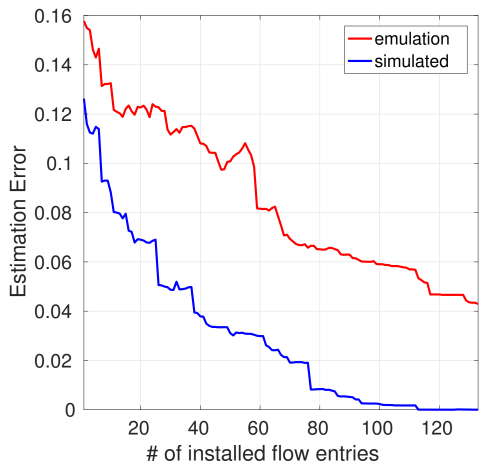


(a) Abilene

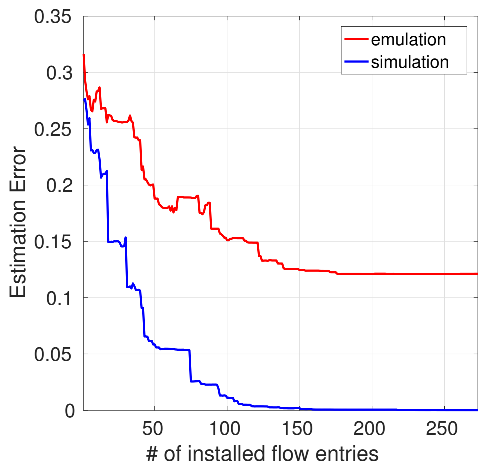


(b) Nobel

Figure 8: Estimation error achieved by CofFE-FAN algorithm and FSBA as a function of the number of installed flow entries in an emulated environment.



(a) Abilene



(b) Nobel

Figure 9: Comparison among the estimation error as a function of the number of installed flow entries achieved by CofFE-FAN algorithm in a simulated environment and in an emulated one.

Table I: CofFE-FAN execution time.

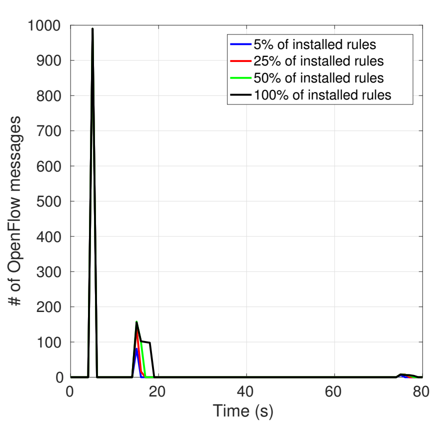
% of flow rules	New rules	Rule Counters
5%	0.02 s ($\sigma = 0.04$)	0.52 s ($\sigma = 0.11$)
25%	0.99 s ($\sigma = 0.12$)	1.05 s ($\sigma = 0.13$)
50%	1.51 s ($\sigma = 0.09$)	2.27 s ($\sigma = 0.14$)
100%	3.28 s ($\sigma = 0.11$)	3.35 s ($\sigma = 0.21$)

of CofFE-FAN algorithm; and ii) number of OpenFlow control messages that must be sent to accomplish such tasks.

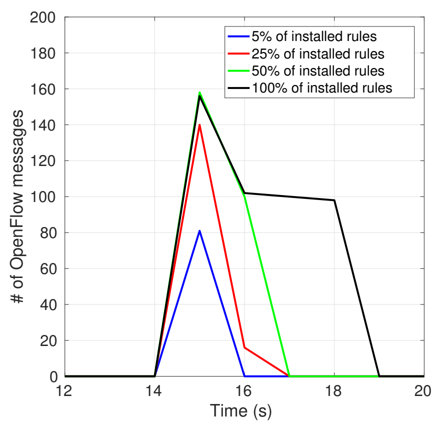
For this analysis, we consider the Abilene topology and four different percentages of new flow rules to install: 5%, 25%, 50%, and 100% of the overall de-aggregation operations. For each of the considered scenarios, there is a different number of nodes the controller must interact with for rule counters collection: 3 nodes in the case of 5%, 8 nodes for 25%, 10 nodes for 50%, and 12 nodes for the full set of flow rules (100%).

Table I shows the CofFE-FAN algorithm execution time as a function of the percentage of installed flow rules for Abilene topology. Due to the fact that the time required by emulation on Mininet depends on several factors such as topology size, number of flows to be sent using iPerf, processes running on the machine, percentage of RAM usage, etc., tests are repeated 10 times to assess values for the standard deviation (σ). The results highlight that increasing the number of installed flow rules leads to higher CofFE-FAN execution time. As in the case of initial rules installation and link counters collection (information not shown in Table I due to a low variability), values reported are small: 3.28 seconds to install the rules assessed by CofFE-FAN when the full set flow rules is considered, and 3.35 seconds to collect their statistics. Remarkably, this last process is the one which requires more time compared with the rest of processes that compose CofFE-FAN.

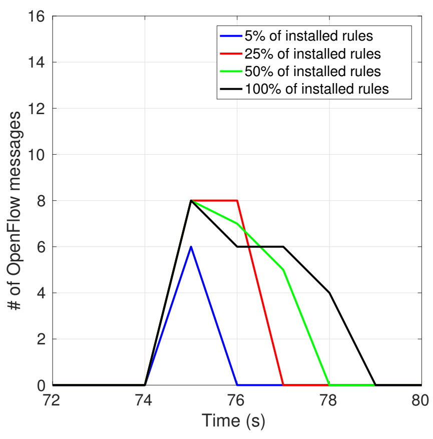
Fig. 10a shows the number of OpenFlow messages per second that are sent during the execution of CofFE-FAN on Abilene network. As first, the installation of the initial rules is carried out on second 5, requiring a total of



(a) Full procedure



(b) Link load collection and new rules installation



(c) Rule counters collection

Figure 10: OpenFlow messages per second versus time in Abilene topology.

990 OpenFlow messages. Since 330 rules must be installed according to Dijkstra and 3 different OpenFlow messages are required to install a single rule (*OFPT_FLOW_MOD*, *OFPT_BARRIER_REQUEST* and *OFPT_BARRIER_REPLY*), 990 messages are sent during the first task of CofFE-FAN.

Five seconds after installing the initial rules, the traffic is sent among every source-destination pair. At second 15, two actions are performed in sequence: i) link load collection; and ii) CofFE-FAN execution with new rules installation. The OpenFlow messages sent by the controller to a specific node for link load statistics are of type *OFPT_STATS_REQUEST*, with the option *OFPST_PORT*. After the message reception, the node returns the requested information inside an *OFPT_STATS_REPLY* message. Since link load statistics for 30 unidirectional links (30 ports) must be collected, a total of 60 OpenFlow messages are required and sent at second 15 (zoom in Fig. 10b).

Upon collecting the statistics related to link loads, CofFE-FAN is executed to assess the new set of rules to be installed. These new rules are created by the controller and sent to the corresponding nodes. Again, the same type of messages as in the case of initial rules are involved (*OFPT_FLOW_MOD*, *OFPT_BARRIER_REQUEST* and *OFPT_BARRIER_REPLY*). As we can see in Fig. 10b, and more precisely in Fig. 11 (where the fixed number of 60 messages at second 15 for link load collection is removed), the time required to install the new rules, highly depends from the percentage of flow rules that is allowed to install (see Table I). Indeed, the difference in time between installing all the flow rules or only the 5% of them is around 3 seconds (black and blue lines).

After the new rules installation, a waiting time $\Delta_t = 60$ seconds is introduced to collect the rule counters. Looking at Fig. 10c, we can see that the controller starts collecting rule counters at second 75 by sending *OFPT_STATS_REQUEST* messages with option *OFPST_FLOW* to the corresponding nodes. It is highlighted that the time required to collect rule counters grows with the percentage of installed flow rules, being 3 times bigger for the case of 100% with respect to the case of 5%. Finally, Fig. 12 shows the aggregated number of OpenFlow messages as a function of time. First, the whole

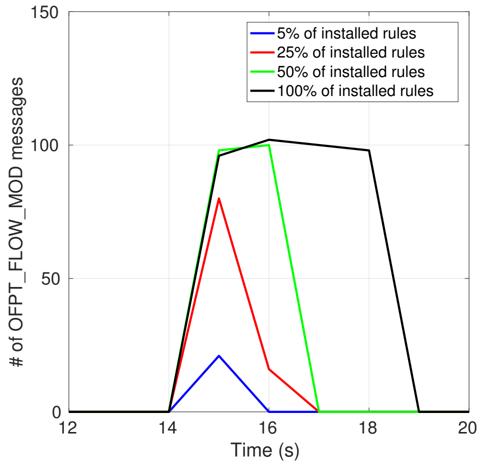


Figure 11: OFPT_FLOW_MOD messages per second during the process of installing the new rules.

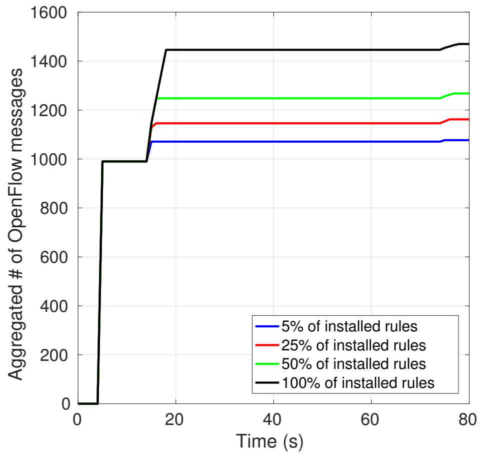


Figure 12: Aggregated number of OpenFlow messages versus time in Abilene topology.

process has a different duration depending on the percentage of installed flow rules. Second, the first two operations of the CofFE-FAN process, i.e., initial rules installation at second 5 and link load collection at second 15, require the same number of OpenFlow messages regardless of the percentage of installed flow rules. Third, the difference in terms of number of messages that must be sent between the controller and the nodes when considering 5% and 100% of installed flow rules, is less than 400 messages. Note that this number is expected to be bigger in large networks. In the same way, an increase in the number of nodes would lead to experience a bigger difference in the time needed to complete the full CofFE-FAN process.

The conducted analysis suggest that a trade off between the duration of the assessment phase, as well as the number of signalling messages, and the final quality of the obtained TM, can be found by properly setting the maximum number of new flow rules to install.

VIII. Conclusions

This paper proposes a mixed measurement and estimation scalable solution to accurately solve the TMA problem on hybrid IP/SDN networks by exploiting the availability of flow rule counters in SDN switches. In particular, the proposed algorithm overcomes other state-of-the-art solutions by i) avoiding the assumption that an IE flow can be measured by installing only one flow rule; ii) considering potential errors collected during the measurement phase; and iii) providing a real implementation on the OpenDaylight controller, which gives information about its effective impact on the network functioning. The performance evaluation shows that our error-tolerant solution is able to assess the TM with a negligible estimation error by only measuring a small percentage of traffic flows. Indeed, the performance analysis of the proposed implementation using the OpenDaylight controller over an emulated network environment, shows that a trade-off between the quality of the assessed TM and its impact on the network in terms of control messages to be sent can be found by properly tuning the number of measured flows.

References

[1] P. Tune and M. Roughan, "Internet Traffic Matrices: A Primer," in *Recent Advances in Networking, Volume 1*, H. Haddadi and O. Bonaventure, Eds. ACM SIGCOMM eBook, 2013.

[2] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "Opentm: traffic matrix estimator for openflow networks," in *International Conference on Passive and Active Network Measurement*. Springer, 2010, pp. 201–210.

[3] M. Malboubi, L. Wang, C. nee Chuah, and P. Sharma, "Intelligent SDN based traffic (de)Aggregation and Measurement Paradigm (iSTAMP)," in *INFOCOM, 2014 Proceedings IEEE*, April 2014, pp. 934–942.

[4] E. Norige, A. X. Liu, and E. Torng, "A ternary unification framework for optimizing tcam-based packet classification systems," *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 657–670, April 2018.

[5] N. Kang, Z. Liu, J. Rexford, and D. Walker, "Optimizing the one big switch abstraction in software-defined networks," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 13–24.

[6] M. Polverini, A. Baiocchi, A. Cianfrani, A. Iacovazzi, and M. Listanti, "The power of sdn to improve the estimation of the isp traffic matrix through the flow spread concept," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 6, pp. 1904–1913, June 2016.

[7] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast Accurate Computation of Large-scale IP Traffic Matrices from Link Loads," *SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 206–217, June 2003.

[8] A. Gunnar, M. Johansson, and T. Telkamp, "Traffic Matrix Estimation on a Large IP Backbone: A Comparison on Real Data," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '04. New York, NY, USA: ACM, 2004, pp. 149–160.

[9] "OpenDaylight," https://www.opendaylight.org/, accessed: 2018-12-04.

[10] B. Claise, "Cisco systems netflow services export version 9," 2004.

[11] P. Phaal, "sflow version 5, jul. 2004," *ieeexplore. ieee. org/xpls/abs_all. jsp*.

[12] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a Better NetFlow," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 245–256, Aug. 2004.

[13] H. Tahaei, R. Salleh, S. Khan, R. Izard, K.-K. R. Choo, and N. B. Anuar, "A multi-objective software defined network traffic measurement," *Measurement*, vol. 95, pp. 317–327, 2017.

[14] J. Suh, T. T. Kwon, C. Dixon, W. Felter, and J. Carter, "Opensample: A low-latency, sampling-based measurement platform for commodity sdn," in *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*. IEEE, 2014, pp. 228–237.

[15] Y. Gong, X. Wang, M. Malboubi, S. Wang, S. Xu, and C.-N. Chuah, "Towards accurate online traffic matrix estimation in software-defined networks," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. ACM, 2015, p. 26.

[16] M. Malboubi, L. Wang, C. N. Chuah, and P. Sharma, "Intelligent sdn based traffic (de)aggregation and measurement paradigm (istamp)," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, April 2014, pp. 934–942.

[17] A. K. Bozkurt, G. Cantali, and G. Gür, "Traffic estimation via kalman filtering under partial information in software-defined networks," in *Proceedings of the Asian Internet Engineering Conference*, ser. AINTEC '17. New York, NY, USA: ACM, 2017, pp. 46–53.

[18] Z. Hu and J. Luo, "Cracking network monitoring in dcns with sdn," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 199–207.

[19] Y. Tian, W. Chen, and C. Lea, "An sdn-based traffic matrix estimation framework," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2018.

[20] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 70–75, Apr. 2014.

[21] J. Galán-Jiménez, "Legacy ip-upgraded sdn nodes tradeoff in energy-efficient hybrid ip/sdn networks," *Computer Communications*, vol. 114, pp. 106 – 123, 2017.

[22] M. Polverini, A. Iacovazzi, A. Cianfrani, A. Baiocchi, and M. Listanti, "Traffic matrix estimation enhanced by sdns nodes in real network topology," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2015, pp. 300–305.

[23] "OpenFlow Switch Specification. Version 1.5.0," https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf, accessed: 2018-12-04.

[24] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "Opennetmon: Network monitoring in openflow software-defined networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–8.

[25] "Mininet: An Instant Virtual Network on your Laptop (or other PC)," http://mininet.org/, accessed: 2018-12-04.

[26] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, April 2007.

[27] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, Mar. 1977.

[28] A. Soule, K. Salamatian, A. Nucci, and N. Taft, "Traffic Matrix Tracking Using Kalman Filters," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 3, pp. 24–31, Dec. 2005.

[29] "iPerf - The TCP, UDP and SCTP network bandwidth measurement tool," https://iperf.fr/, accessed: 2018-12-04.