Title

OpenMeasure: Adaptive Flow Measurement and Inference with Online Learning in SDN

Authors
Liu, Chang
Malboubi, Mehdi
Chuah, Chen-Nee

Publication Date
2016-04-01

Peer reviewed

OpenMeasure: Adaptive Flow Measurement & Inference with Online Learning in SDN

Chang Liu, Mehdi Malboubi, Chen-Nee Chuah
Dept. of Electrical & Computer Engineering
University of California, Davis, USA.
{cchliu, mmalboubi, chuah}@ucdavis.edu

***Abstract*—Accurate and efficient network-wide traffic measurement is crucial for network management. Recently, Software-defined networking (SDN) has opened up new opportunities in network measurement and inference. In this work, we demonstrate an efficient flow measurement and inference framework which performs adaptive measurement with online learning. Using the reprogrammability of SDN, we assist network inference with online learning predictions and dynamically update the measurement rules network-wide to track and measure the most informative flows. To best utilize the available measurement resources, we leverage the SDN controller (with its global view) to optimally place flow monitoring rules across network switches. Using real-world data, we show that our measurement framework achieves high performance in both estimating the traffic matrix and identifying hierarchical heavy hitters.**

I. Introduction

Fine-grained traffic size information provides an essential input for many network design and operation tasks, such as capacity planning, network provisioning, load balancing, and anomaly detection [1]–[3]. Flow size can be measured directly or inferred (indirectly) from sampled statistics. In large-scale networks, direct flow measurement can be challenging due to the exploding traffic volume, limited monitoring resources and the prohibitively large overhead imposed on the network components. An alternate approach is estimating the traffic matrix (TM) from a set of aggregated/end-to-end measurements using network inference techniques.

The emerging software defined networking (SDN) paradigm decouples network control (control plane) from forwarding functions (data plane), enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services [4]. The control logic is moved to an external entity, the SDN controller. The logically centralized controller has a global view of network and can dynamically configure the forwarding rules in managed flow tables. In addition, it can also obtain flow statistics from reading the counters of the switch TCAM rules.

SDN opens up many new opportunities for addressing network measurement and inference problems. On one hand, it can augment network inference with multi-resolution monitoring and online learning. On the other hand, with the global view of the network, the controller can optimize the network-wide allocation of measurement resources. We could dynamically determine what/when/where to measure.

While there have been many studies on leveraging SDN in network measurement and inference, the most relevant work is iSTAMP [5]. However, iSTAMP does not take routing and flow aggregation feasibility into account when designing optimal flow aggregates and only focuses on single-switch scenario. While [6] extended iSTAMP framework to multi-switch scenario, the discussion about how to continuously identify large flows and update measurements over time is missing. Different from [5] and [6], in our work, we propose a framework we refer to as *OpenMeasure* which leverages continuous online learning techniques to perform network-wide adaptive flow measurement and inference.

Similar to iSTAMP, OpenMeasure consists of an intelligent flow sampling and network inference engine residing on the SDN controller. However, instead of optimizing the aggregation matrix, OpenMeasure assumes that the aggregation matrix is given based on the underlying routing and flow aggregation rules. As mentioned earlier, our framework employs an online learning algorithm to determine the most informative flows for sampling. By leveraging the global view of SDN controller, it identifies the available monitoring resources and intelligently places flow sampling rules in selected SDN switches. Furthermore, this framework is light-weight, compatible with hybrid SDN networks, relies on current capabilities of OpenFlow (OF) switches, and does not impact network routing functions.

To summarize, the contributions of our work are three-fold:

- We propose *OpenMeasure*, a network-wide adaptive flow measurement and inference framework with continuous learning capability. We propose two online learning algorithms for designing the adaptive flow measurement rules: one algorithm is based on weighted linear prediction and the other adopts the strategy used in multi-armed bandit (MAB) problems [7].
- Leveraging SDN controller's global view of network, we optimize the allocation of flow monitoring rules among multiple OF switches to increase measurement accuracy. We formulate the problem mathematically and proposes two light-weight heuristic rule allocation algorithms.
- We evaluate the performance of OpenMeasure in traffic matrix (TM) estimation as well as hierarchical heavy hitter (HHH) identification. We demonstrate the benefit of *continuous learning* and *update* of measurement rules, which is absent in [6]. We also implemented OpenMeasure on GENI [8] testbed to demonstrate the practical

effectiveness and feasibility of our framework [9].

The rest of the paper is organized as follows. Section II presents an overview of OpenMeasure followed by details of key components in the framework. Section III discusses applications of our measurement framework. Section IV evaluates OpenMeasure on a number of performance metrics. Section V summarizes the related work and we conclude in section VI.

II. Methodology

Without assuming complex functions on OF switches, our framework relies on the simple match-and-count rules installed at the switch dataplane and managed by the controller. The design of the match-and-count rules is critical in measurement.

A. Overview

Since the transition to SDN requires simultaneous support of SDN and legacy equipment, OpenMeasure assumes a hybrid SDN network, where SDN-enabled and non-SDN routers/switches coexist in the network. All the SDN-enabled routers/switches are managed by a logically-centralized controller and are pre-populated with local routing rules. Aside from these routing entries, a handful of TCAM entries are available for implementing measurement/monitoring rules.

OpenMeasure contains three components. The first component identifies the most informative flows from online learning and prepares a set of "candidate" rules to be installed. The second component dynamically determines where/what rules are to be installed network-wide using the controller's global view. Finally, the third component periodically pulls traffic statistics from switches (both TCAM counters from OF switches and SNMP link loads) and deploys existing inference techniques to estimate the traffic matrix or to support other monitoring applications (e.g., hierarchical heavy hitter identification).

B. Rule design

1) Initial set of rules: Due to the limited number of TCAM entries, it is infeasible to maintain a counter per fine-grained flow. The routing rules are aggregated based on the destination prefixes. However, coarse aggregation may introduce large flow estimation error. Aside from the routing entries, new measurement rules can be installed in the available TCAM entries by offloading subsets of flows from the original routing rules. However these subsets of flows can not be arbitrarily selected because 1) the current implementation of TCAMs limits that only flows with the same source/destination ip prefixes can be aggregated in one TCAM rule; and 2) global routing can not be disrupted.

OpenMeasure starts with collecting coarse-grained statistics. It adopts the Maximum Load Rule First (MLRF) method [6] to populate the initial measurement rules. The load of a rule here is defined as the number of flows matching the rule in a SDN switch. MLRF starts from the rule with the maximum load and generates a new rule with a longer source IP prefix and a higher priority to offload half of the flows from the selected rule. Initial estimate of per-flow sizes are obtained from measurements generated by MLRF method.

2) Adaptive mechanism with online learning: Identifying/Measuring the most informative/rewarding flows is important for improving TM estimation accuracy [5]. We propose that without altering pre-populated routing entries, OpenMeasure utilizes the available TCAM entries to sample (i.e., directly measure) a set of large flows. In the beginning, these large flows are identified based on the initial estimates of traffic matrix using only aggregate counts. Subsequently, OpenMeasure pulls statistics from both aggregate counters and sampled flows. Upon receiving the new statistics, OpenMeasure runs online learning algorithms to determine an updated set of large flows, and updates flow tables accordingly.

In general, it is challenging to identify large flows for future time intervals, since per-flow sizes are unknown a priori. We propose two online learning algorithms below.

a) Weighted Linear Prediction (WLP): The Weighted Linear Prediction method uses previous samples to estimate or predict a future value. It attempts to predict the per-flow size of the next measurement. Specifically, we maintain a list of n records for each flow. The predicted size of flow fl at time t_n is in (1):

$$v_p^{fl} = \lambda(v_n^{fl} + \sum_{i=1}^{n-1} e^{-(t_n - t_i)} v_i^{fl}) \tag{1}$$

Weighted Linear Prediction assumes a exponential decaying impact over time. λ is a scaling factor. Based on the predicted per-flow sizes obtained, WLP ranks all the flows in descending order of flow sizes and based this result, OpenMeasure selects a set of large flows to track and measure.

b) Modified Upper Confidence Bound (MUCB) Prediction (MUCBP): In MUCBP, we take the flow size of a directly sampled flow as the reward for sampling this flow, then our goal is to maximize the total reward obtained from all the available TCAMs over time. The key point is to balance between acquiring new information (exploration) and capitalizing on the information available so far (exploitation) [7]. Specifically, the predicted reward of directly measuring flow fl computed at time t is in (2):

$$I_{fl}^t = \alpha \overline{x}_{fl} + \sqrt{\frac{2\ln(n)}{n_{fl}}} \tag{2}$$

where α is a scaling factor, $\overline{x}_{fl}$ is the average flow size for flow fl, n_{fl} is the number of times flow fl has been directly measured so far and n is the overall number of direct measurements done so far. The first term, sample mean, favors exploitation while the second term, confidence bound, favors exploration [7].

Different from [5], we obtain the initial system dynamics from measurements generated by MLRF method instead of observing each flow [5], which is not feasible. MUCBP ranks all flows in descending order of reward and based on this result, OpenMeasure selects a set of large flows to track and measure.

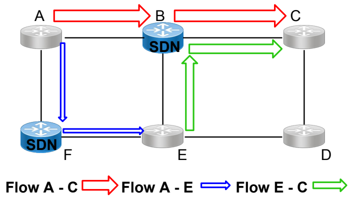

Fig. 1: Illustration of spatial mismatch in rule allocation

C. Rule placement

Once the set of large flows to be monitored is identified, we need to determine where (which switch) to measure what (which flows). Since the SDN controller has the complete network topology and routing path for each flow, we have the opportunity to optimize monitoring rules placement network-wide.

Ideally we would like to use the K available TCAM entries to directly measure the K most informative flows. However, it is not always as straight-forward in a hybrid multi-switch scenario. A simple example in Fig. 1 shows the spatial mismatch between the available TCAM entries and the route of large flows. There are three flows in Fig. 1: *A-C, A-E, E-C*. Flow size order: $A-C > E-C > A-E$. Assume both SDN switches S_B and S_F have one available TCAM entry. Apparently, although there are two TCAMs available in the network, it is not feasible to sample the two largest flows due to spatial mismatch.

The rule placement problem can be described as follows. Given a network G with V set of OF switches and a set of flows to be measured F, we need to find the best mapping from F to V under the following constraints: 1) Flows assigned to OF switch $s \in V$ should pass switch s in their routes. 2) The number of assigned rules in $s \in V$ should not exceed the number of TCAM entries available for monitoring tasks. 3) In order to save TCAM entries, the same flow should not be sampled more than once. The objective is to maximize the sum of the reward/per-flow size obtained from all the available TCAMs in the network. The problem could be formulated into an integer linear program (ILP) as shown in (3):

$$
\begin{aligned}
\underset{D}{maximize} \quad & W(D) = \sum_{i=1}^{|V|}\sum_{j=1}^{|F|} P_{f_j} d_{ij} \\
\text{subject to} \quad & d_{ij} \in \{0,1\},\ \forall i = 1 \cdots |V|,\ j = 1 \cdots |F| \\
& d_{ij} \le h_{ij},\ \forall i = 1 \cdots |V|,\ j = 1 \cdots |F| \\
& \sum_{j=1}^{|F|} d_{ij} \le K_i,\ \forall s_i \in V \\
& \sum_{i=1}^{|V|} d_{ij} \le 1,\ \forall j = 1 \cdots |F|
\end{aligned}
\tag{3}
$$

where P_{f_j} is the predicted flow size in (1) with WLP learning algorithm and the predicted reward in (2) with MUCBP learning algorithm, for a sampling flow $f_j \in F$. Binary matrix $D = \{d_{ij}\}$ denotes a feasible rule placement solution. The element d_{ij} indicates whether flow f_j is directly measured at OF switch s_i. K_i is the number of available TCAMs at switch s_i. Binary matrix $H = \{h_{ij}\}$ denotes the global routing matrix which is known at the logically centralized controller. The element h_{ij} indicates whether flow f_j passes OF switch s_i in its routing path.

We could get the optimal rule placement solution by solving this ILP with the Gurobi [10] solver. However, as the size of the network increases, the computational complexity increases exponentially. As the optimal solution may be time consuming, we propose two heuristic algorithms with linear computational complexity, which can efficiently find a feasible and near-optimal allocation solution in practical cases.

Algorithm 1 LastHop placement algorithm

```
procedure LASTHOP_RULE_PLACEMENT($G, V, SF, K$)
    for each $fl \in SF$ do
        gather all switches in $V$ that cover $fl$ as $S^{fl}$
        sort $S^{fl}$ in order from dst to src as $SS^{fl}$
        for each OF switch $s \in SS^{fl}$ do
            if $K_s > 0$ then
                assign $fl$ to switch $s$
                $K_s = K_s - 1$
                break
```

Algorithm 2 Greedy placement algorithm

```
procedure GREEDY_RULE_PLACEMENT($G, V, SF, K$)
    for each switch $s \in V$ do
        compute large flow coverage count $C_s$
    for each $fl \in SF$ do
        gather all switches in $V$ that cover $fl$ as $S^{fl}$
        sort $S^{fl}$ in descending order of $K_s$ as $SS^{fl}$
        select switch $s$ with maximum $K_s$
        if there is a tie then
            select switch $s$ with minimum $C_s$ from the tie
switch list
        if $K_s > 0$ then
            assign $fl$ to switch $s$
            $K_s = K_s - 1$
```

Algorithm 1 and Algorithm 2 show the details of the two heuristic algorithms. We sort the set of flows selected to be monitored, in descending order of per-flow size/reward based on the outcome of online learning algorithms, denoted as SF, and take it as an input for the rule placement algorithms. We map each flow in SF to a set of candidate switches that can monitor it, denoted as S^{fl}. For each OF switch $s \in V$, we keep a counter for the number of available TCAM entries it has, K_s. For both heuristics, we start with flows ranking highest (largest flows) and iteratively assign flows to switches.

In LastHop rule placement algorithm (LRP), we sort the candidate switches $s \in S^{fl}$ in the order of positions in route path from destination to source. We first check if the last

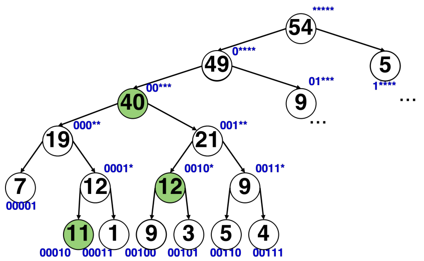

Fig. 2: HHHs in an example prefix tree of source IP addresses. The number at each node represent the traffic volume sent by the IP prefix during a certain time interval. Threshold is set to 10 here. All the HHHs are shaded in green color. Node 001** is not an HHH because more than half of its large volume comes from its HHH descendant 0010*.

OF switch in the flow path has available TCAMs to count flow. If not, it moves to a prior OF switch. The process is repeated until it finds a spot to monitor or it fails to find any available TCAMs among all the candidate switches. In Greedy Rule Placement algorithm (GRP), we keep an extra counter for each OF switch $s \in V$ for the number of large flows it can monitor, C_s. For each flow, we greedily assign the flow to the switch $s \in S^{fl}$ with the current maximum available capacity. If there is a tie, we choose the one with the smallest C_s. It means that we save switches which can cover more large flows for later assignment. Assume n is the the number of flows, the computational complexity for both heuristic algorithms is $\mathcal{O}(n)$.

III. Example Application

OpenMeasure periodically reads traffic counters from switches and estimates the traffic matrix. Our measurement framework is useful for many applications that use these flow size estimates, such as, monitoring normal traffic pattern, detecting large traffic changes, and identifying hierarchical heavy hitters (HHHs).

In this work, we take the HHH identification problem as a case study to show the effectiveness of our framework and understand the benefit of adaptive measurement with SDN-enabled online learning. Detecting HHHs is important for a number of security applications, including pinpointing denial-of-service (DoS) attacks and DDoS attacks [11], discovering worms [12] and other anomalies. We focus on detecting HHHs in the hierarchical domain of source IP prefixes. To find HHHs, we build a prefix tree of source IP addresses for each destination IP prefix. HHHs are the longest IP prefixes whose aggregated traffic volume is larger than a user-specified threshold, after excluding the contribution of any HHH descendant [13]. Fig. 2 shows an example in detail.

IV. Performance Evaluation

In order to evaluate the performance of the adaptive measurement with SDN-enabled online learning, we compare it with 1) Random Scheme (RS), where we randomly assign a subset of flows to each switch to monitor; 2) Static Aggregation (SA), where we use the initial set of rules (fixed grouping of flows) during the entire period; 3) LFF proposed in [6]. LFF identifies the set of large flows from the statistics obtained from MLRF method in the first phase, and measures a static set of large flows during the entire counting period. We build a simulator to evaluate different measurement strategies using the topology and traffic traces from a publicly available dataset from the Geant network [14]. We evaluate OpenMeasure from two aspects: the accuracy for TM estimation and the performance for HHH identification.

A. Trace-driven simulation

We perform our simulations using topology and traffic traces from the Geant network (23 nodes and 37 links) [14]. This publicly available dataset contains traffic collected for each 15 minutes interval over a time range of 4 months. We randomly select a week's data as the ground truth in our experiment. The data provides traffic sizes between any two nodes in the topology. In order to get fine-grained traffic matrices, we randomly assign a number of IP prefixes to each node. Each IP prefix is associated with a weight. The fine-grained flow size between any pair of IP prefixes is calculated using (4):

$$S_{f_i} = S_{sd} * \frac{w_{src\_prefix}}{\sum\limits_{prefix \in src\_node} w_{prefix}} * \frac{w_{dst\_prefix}}{\sum\limits_{prefix \in dst\_node} w_{prefix}} \tag{4}$$

Since we consider a hybrid SDN network, we assume only a subset of nodes are OF switches. The nodes with larger flow coverage sets are more likely to be SDN-enabled. In our case, we choose 6/23 nodes to be OF switches and we assume the number of TCAM entries is the same for all OF switches. The routing entries are generated based on shortest-path routing.

B. Accuracy for TM estimation

Normalized Mean Square Error (NMSE): We use the metric NMSE to examine the accuracy of traffic matrix estimation. NMSE is defined in (5).

$$NMSE = \frac{1}{T_c} \sum_{t=1}^{T_c} \frac{\| X^t - \hat{X}^t \|_2}{\| X^t \|_2} \tag{5}$$

where X^t is the true traffic matrix at time slot t and $\hat{X}^t$ is the estimated traffic matrix. T_c is the total number of time intervals measured. We compute the NMSE under different measurement strategies and compare OpenMeasure with Random Scheme (RS), Static Aggregation (SA), and LFF [6].

First, we evaluate the performance of different rule placement algorithms with the same online learning algorithms. Fig. 3 plots the average NMSE obtained using LRP, GRP and optimal rule placement solution (by directly solving the ILP). The scheme that uses Weighted Linear Prediction (WLP) online learning algorithm in rule design and Greedy rule placement (GRP) algorithm in rule placement is denoted as OpenMeasure(WLP+GRP). The aggregation ratio r is defined

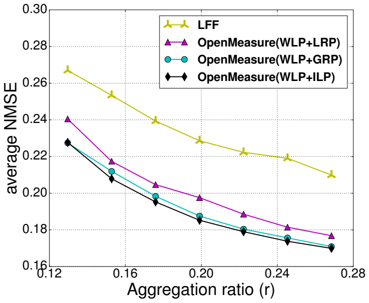

Fig. 3: NMSE under different rule placement algorithms

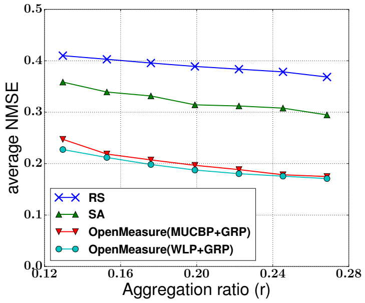

Fig. 4: NMSE with varying r

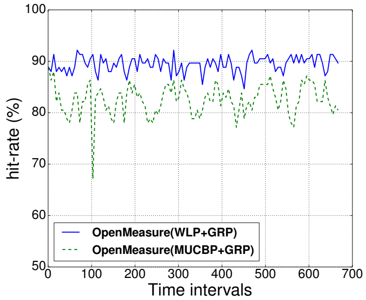

Fig. 5: Hit-rate of different online learning algorithms, $K = 120$

as the total number of TCAM entries used over the number of flows in the network. From Fig. 3, we can observe that the two heuristic algorithms achieve estimation accuracy that is very close to that of the optimal solution, showing the effectiveness of our proposed heuristics with continuous learning. Among the two heuristic algorithms, Greedy rule placement (GRP) algorithm performs slightly better than LastHop rule placement (LRP) algorithm. For simplicity, we use the results from GRP to represent the performance of OpenMeasure in the rest of the discussion.

Now we compare OpenMeasure with LFF [6] in Fig. 3. Both OpenMeasure(WLP+ILP) and LFF optimally solves the rule placement problem. However LFF identifies/measures the set of large flows based on “one-time” learning, while OpenMeasure performs continuous learning to adaptively update measurements to track large flows online. The gap between the curve of OpenMeasure (WLP+ILP) and that of LFF shows the advantage of continuous learning.

Fig. 4 shows the average NMSE obtained using RS, SA, and OpenMeasure, with different aggregation ratio r. We can see that as expected, the average NMSE decreases as r increases, since more measurements are available. Moreover, we can clearly observe that OpenMeasure reduces estimation error drastically, showing that adaptive measurement with SDN-enabled learning could greatly improve TM estimation accuracy compared with measurement without learning. The conclusion holds true as aggregation ratio r varies. Now let us focus on the two OpenMeasure curves with different online learning algorithms. WLP performs slightly better than MUCBP. The result could be further explained in Fig. 5. The idea of OpenMeasure is to utilize the available TCAMs to directly measure large flows. The key challenge of online learning algorithms is to accurately identify large flows. We define hit-rate as the true large flows out of the K largest flows reported by online learning algorithms over K, where K is the total number of TCAM entries available for new measurements in the network. Fig. 5 shows the hit-rate of the two learning algorithms over counting periods. From Fig. 5, we can observe that WLP is more accurate in terms of identifying large flows compared with MUCBP. This arises from the fact that the Geant traffic data is relatively steady over time in a resolution of 15 minutes.

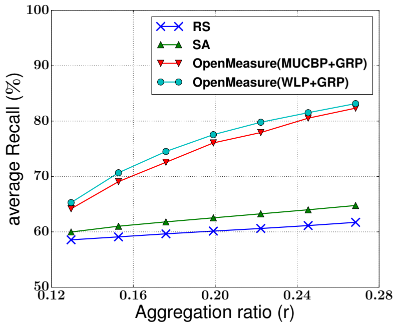

Fig. 6: Recall with varying r

C. HHH identification

Recall & Precision: To quantify the performance of HHH detection, we use two metrics—recall and precision [15]. Recall is the total number of true HHHs detected over the real number of HHHs in the ground truth in each time interval. The higher the recall, the higher the probability of detection. Precision is the total number of true HHHs detected over the total number of HHHs reported in each time interval. The higher the precision, the lower the false positives [13].

Fig. 6 and Fig. 7 show the results for HHH identification. In both figures, we can see that OpenMeasure consistently outperforms the random scheme and static aggregation with more than 10% performance gain. And as r increases (more available TCAMs), the performance gain increases. WLP performs slightly better than MUCBP due to the reason discussed above. Our results demonstrate that a higher accuracy could be achieved in detecting HHHs using adaptive measurement with SDN-enabled online learning.

D. Practical Implementation on GENI testbed

GENI is a distributed virtual laboratory for at-scale experiments in network science, services and security [8]. We performed a practical implemention and evaluation of our OpenMeasure framework on this testbed. Specifically, we (1) created the Geant topology on GENI testbed, (2) installed and configured Open vSwitch [16] on switch nodes to simulate the function of OF switches, and (3) developed an OpenMeasure prototype using the POX OpenFlow API [17]. A demo and a poster [9] based on the preliminary result of this work were

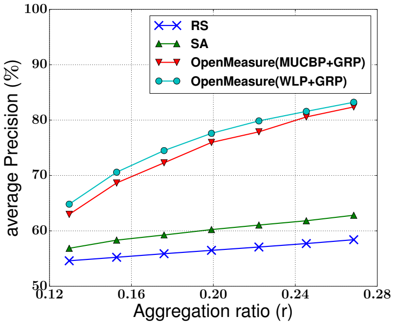

Fig. 7: Precision with varying r

shown in GEC23 and GENI NICE workshop(in 2015). This implementation study demonstrates the practical effectiveness and feasibility of our framework.

V. Related Work

Our work builds on top of the existing proposals of SDN. For TM measurement and inference, OpenTM [18] and OpenNetMon [19] measure the traffic matrix by keeping track of statistics for each flow. These per-flow based measurement solutions do not scale well with the increase of traffic size and impose heavy overhead on the network. The work most relevant to us are iSTAMP [5] and [6]. iSTAMP faces aggregation feasibility issues in practical implementation and only focuses on a single OF switch. Gong et al. [6] extend the iSTAMP framework to the multi-switch case, but do not explore the benefits of continuous online learning in flow measurement and inference like we do. A recent work [20] proposes using TCAM entries to assist network tomography in data center networks.

On the other hand, ProgME [21] proposes a programmable traffic measurement architecture to measure any arbitrary set of flows. OpenSketch [22] proposes using sketches for different measurement tasks. However they rely on specialized hardware support on switches. Jose et al. [13] focuses on detecting hierarchical heavy hitters on a single OF switch. [23] proposes OpenWatch for measuring flows under applications of anomaly detection.

VI. Conclusion and Discussion

This paper presents OpenMeasure, an adaptive fine-grained flow measurement and inference framework, which takes advantage of online learning and global optimization (in rule placement) enabled in SDN. It employs a learning based dynamic adjustment scheme to continuously track and measure the most informative flows. Aside from the two learning algorithms we presented in this work, more advanced machine learning approaches could be applied in our framework to improve prediction accuracy. With the SDN controller's global view, OpenMeasure optimizes the measurement resource allocation across the network. We also propose two light-weight heuristic rule placement algorithms, which achieve close-to-optimal performance but exhibit much smaller computational complexity. Lastly, using topology and traces from a real ISP network, we demonstrate the benefit of *continuous learning* and *update* of measurement rules in both TM estimation and HHH identification applications.

References

[1] M. Roughan, M. Thorup, and Y. Zhang, "Traffic engineering with estimated traffic matrices," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. ACM, 2003, pp. 248–258.

[2] T. Benson, A. Anand, A. Akella, and M. Zhang, "Microte: Fine grained traffic engineering for data centers," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*. ACM, 2011, p. 8.

[3] A. Soule, K. Salamatian, and N. Taft, "Combining filtering and statistical methods for anomaly detection," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. USENIX Association, 2005, pp. 31–31.

[4] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

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

[6] 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.

[7] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[8] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "Geni: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, 2014.

[9] C. Liu, S.-M. Peng, M. Malboubi, C.-N. Chuah, M. Bishop, and B. Yoo, "Distributed iceberg detection with sdn-enabled online learning," *demo presented at 23rd GENI Engineering Conference*, 2015.

[10] "Gurobi optimizer: http://www.gurobi.com/."

[11] P. E. Ayres, H. Sun, H. J. Chao, and W. C. Lau, "Alpi: A ddos defense system for high-speed networks," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 10, pp. 1864–1876, 2006.

[12] G. Huang, A. Lall, C.-N. Chuah, and J. Xu, "Uncovering global icebergs in distributed streams: Results and implications," *Journal of Network and Systems Management*, vol. 19, no. 1, pp. 84–110, 2011.

[13] L. Jose, M. Yu, and J. Rexford, "Online measurement of large traffic aggregates on commodity switches," in *Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services–Hot-ICE. USENIX*, 2011.

[14] "GEANT traffic: https://totem.info.ucl.ac.be/dataset.html."

[15] G. Cormode and M. Hadjieleftheriou, "Methods for finding frequent items in data streams," *The VLDB Journal*, vol. 19, no. 1, pp. 3–20, 2010.

[16] "Open vSwitch: http://openvswitch.org/."

[17] "POX: https://openflow.stanford.edu/display/onl/pox+wiki."

[18] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "Opentm: traffic matrix estimator for openflow networks," in *Passive and active measurement*. Springer, 2010, pp. 201–210.

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

[20] Z. Hu and J. Luo, "Cracking network monitoring in dcns with sdn," in *Proc. IEEE INFOCOM*, 2015.

[21] L. Yuan, C.-N. Chuah, and P. Mohapatra, "Progme: towards programmable network measurement," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 1, pp. 115–128, 2011.

[22] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch." in *NSDI*, vol. 13, 2013, pp. 29–42.

[23] Y. Zhang, "An adaptive flow counting method for anomaly detection in sdn," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 25–30.