Rhaban Hark, Khalil Tounsi, Amr Rizk, and Ralf Steinmetz. *Decentralized Collaborative Flow Monitoring in Distributed SDN Control-Planes (accepted for publication)* In: Proceedings of the Conference on Networked Systems (NetSys'19), p: 1–8, IEEE, March 2019.

Decentralized Collaborative Flow Monitoring in Distributed SDN Control-Planes

Rhaban Hark, Khalil Tounsi, Amr Rizk, Ralf Steinmetz
Multimedia Communications Lab
Technische Universität Darmstadt
Darmstadt, Germany
{*first.last*}@kom.tu-darmstadt.de

***Abstract*—The Software-defined Networking paradigm became widely accepted in academia as well as the industry in the last decade. The logically centralized control-plane provides simple mechanisms to support efficient monitoring as proposed in a plethora of works recently. However, existing works simplify the control-plane to a single entity whereas it is proven that the control-plane must be implemented physically distributed.**

To this end, we lately proposed DISTTM, a system to increase the efficiency while monitoring flows in distributed control-planes by eliminating redundant measurements of shared flows. The system is based on a centralized coordinator that introduces a single-point-of-failure and is vulnerable to network partitioning. To overcome this shortage, we propose a distributed algorithm to assign redundant measurement responsibilities to single controllers and share information among controllers in the work in hand. In an evaluation, we show that we decrease the number of measurements strongly and improve the load fairness among controllers through the collaboration of networks.

***Index Terms*—Software-defined Networking, Flow Monitoring, Distributed Algorithms, Distributed Control Plane**

I. INTRODUCTION

Since the Software-defined Networking (SDN) [1] paradigm was proposed in 2008 from the Stanford University, it gained popularity in academia and finds its way into the industry. SDN relives the idea of centralized control, placed on the control-plane, of the forwarding network, denoted data-plane. Thereby, it provides a variety of advantages such as simple, flexible network management and open-standard hardware, combined with new techniques, e.g. to monitor the network using easily accessible flow-level counter. In future networks, network state monitoring plays a key role reasoned by the increasing demand on adaptability. Nevertheless, as monitoring has only passive impact on the productivity of a network, it is required to be non-invasive and economic with respect to the available resources.

A variety of works propose intelligent mechanisms to efficiently monitor Software-defined Networks on the different SDN layers. Improvements regarding the measurement on the data-plane [2]–[4], the process of collecting it [5], [6] and general purpose monitoring frameworks [7], [8] were made. The mentioned approaches focus on the work within a simplified version of SDN, namely with a single controller. However, a control-plane consisting of a single controller introduces a single point of failure with all its disadvantages regarding failsafety and scalability and is, therefore, not a feasible solution for almost any type of network.

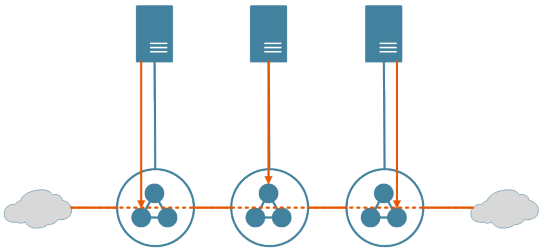


Fig. 1: The flow in the data-plane is monitored redundantly in each network part controlled by different controllers of a distributed control-plane.

The control-plane within a domain that holds a logically centralized controller must be implemented using a physically distributed network of collaborating controllers [9]–[13]. Subsequently, not a single controller controls the full network but each part of the network is controlled by a different controller of the distributed control-plane. In such scenarios, flows usually traverse not only one part of the network, but multiple adjacent networks. As sketched in Figure 1 the existing monitoring approaches do not cover such scenarios and leave this responsible to the control-plane implementation.

In a previous work, we propose DISTTM [14], a distributed system to eliminate the redundancy in such situations. To this end, DISTTM defines the behavior of controllers within a distributed control-plane and their interaction to assign monitoring responsibilities to a single controller in contrast to monitoring in over and over again. DISTTM, furthermore, provides different schemes to ensure load fairness based on different strategies such as, e.g., the ratio of flows monitored by a controller. We were able to show a significant reduction of monitoring costs compared to a non-cooperative solution with redundancy.

However, DISTTM introduces a centralized coordinator that is selected from the set of participating controllers. The coordinator gathers monitoring interests and calculates which controller is responsible for a measurement based on the fairness strategy. By using a central coordinator, the system becomes vulnerabel to failures of this coordinator and network partitioning. Subsequently, in this work we propose an algorithm

Rhaban Hark, Khalil Tounsi, Amr Rizk, and Ralf Steinmetz. *Decentralized Collaborative Flow Monitoring in Distributed SDN Control-Planes (accepted for publication)* In: Proceedings of the Conference on Networked Systems (NetSys'19), p: 1–8, IEEE, March 2019.

for decentralized assignment of monitoring responsibilities in scenarios where flows traverse multiple networks controlled by a distributed control-plane. By analogy with DISTTM we measure the traffic of all flows in the network to fill a traffic matrix per controller. The proposed algorithm lets controllers advertise for a flow monitoring responsibility after waiting a generated backoff time. The first controller that distributes its advertisement wins the election and takes the responsibility. We use the generated backoff to ensure fairness among the controllers, a controller generates a rather high waiting time when its portion of responsibilities is high compared to the other controllers. As a matter of course that a high backoff waiting time leads to a lower responsibility of winning an election. Furthermore, we add a mechanism to avoid collisions in terms of simultaneously sent advertisements.

In the evaluation, we show a reduction of monitoring costs by tracking the number of measurements plus the overhead generated from the coordination and statistic sharing. Furthermore, we show in the evaluation an improvement of the load fairness among controllers.

The remainder of the paper is structured as follows: Section II discusses background and works that are relevant for the work in hand on monitoring in SDN, particularly the estimation of traffic matrices as comprised in this work, and works on collaboration among networks. Section III briefly recaps the proposed DISTTM system. Further on, Section IV presents the design of the algorithm developed in this work. Finally, Section V shows the evaluation of the proposed approach while Section VI concludes the paper.

II. Background and Related Work

In this section, we discuss basic backgrounds and works related to our paper. First, we give a brief overview on monitoring approaches designed for the context of SDN. Afterwards, we particularly focus on traffic matrix estimation approaches while the scope is opened to legacy networks as well. Finally, the last subsection presents works towards collaboration of networks and therewith consensus protocols.

A. Monitoring in Software-defined Networks

Software-defined Networks provides new techniques for the control to access meta information from the forwarding network. The centralized controller can, e.g., use byte/packet/lifetime on flow or port level using protocols such as the widely accepted OpenFlow [15] protocol. Works, such as OPENSKETCH [2], DCM [3], and FLEXAM [4] provide custom data-plane implementations to further extend the possibilities to access meta information efficiently. With the development of P4 [16] even more sophisticated techniques can be developed and leveraged. Using the mentioned techniques eases the monitoring process and reduces the overhead since it is directly integrated in the control-loop between the controller and the network.

Subsequently, a plethora of works propose approaches how to monitor equipped with such techniques efficiently in the SDN context. Yu et al. [5] propose FLOWSENSE, a system that utilizes OpenFlow specific control-messages, namely *FlowRemoved* messages, dispatched when flows are removed to capture the bandwidth utilization with zero additional costs. Zhang [6] develops a mechanism denoted OPENWATCH to dynamically adapt flow rules so that it allows timely and spatial zooming and aggregation to optimally gain accurate meta-information of traffic. Despite these exemplary specific approaches, a number of more generic monitoring frameworks were proposed [7], [8].

With regard to this work, Yu et al. [3] tackle the problem of redundant flow measurement due to coarse grained rule aggregation. They develop a mechanism using two-stages Bloom filters that can monitor individual flows without requiring a single rule for each flow. In contrast to our work, Yu et al. focus on the elimination of redundant measurements among switches within a single network and not on controller level.

B. Network Traffic Matrix Estimation

A traffic matrix is a collection of primitive metrics for each ingress-to-egress node pair of a network [17], [18]. The primitive metric stored within the matrix's cells can vary from traffic amount to end-to-end loss. Potential use cases include usually long-term management tasks such as capacity planning, network provisioning, and load balancing policies. However, traffic matrices can also support security tasks like, e.g. anomaly detection [17].

Due to the high amount of information captured in traffic matrices, traditionally monitors use samples and statistical assumptions to estimate the matrix [19], [20]. However, with the advent of SDN and the infrastructure it provides, direct measurements become feasible. OPENTM [21] proposes a traffic matrix estimator for OpenFlow networks. In this work, the system keeps track of active flows using input from the routing application. For each active flow, it selects a switch on its path to query the desired statistic. The authors propose different metrics to select the measurement point to optimize different targets such as a balanced load among the switches or the highest accuracy.

The DISTTM [14] system provides an extension of OPENTM for traffic matrix generation in SDNs and eliminates redundant measurements through coordination among controllers of adjacent networks. DISTTM is the basis of this work and subsequently discussed in detail in the next section.

C. Network Collaboration and Consensus Protocols

As single controllers are not capable of managing large-scale networks, there is a consensus that the logically centralized SDN control-plane must be implemented physically distributed. To this end, a number of works develop infrastructures and architectures to build distributed control-planes [9]–[13]. The works concentrate on different aspects such as consistent common information bases [9], communication channels among controllers [12], or hierarchical infrastructures to reduce load on single entities [13]. Only Phemius et al. [11] mention monitoring as part of the architecture in their system

Rhaban Hark, Khalil Tounsi, Amr Rizk, and Ralf Steinmetz. *Decentralized Collaborative Flow Monitoring in Distributed SDN Control-Planes (accepted for publication)* In: Proceedings of the Conference on Networked Systems (NetSys'19), p: 1–8, IEEE, March 2019.

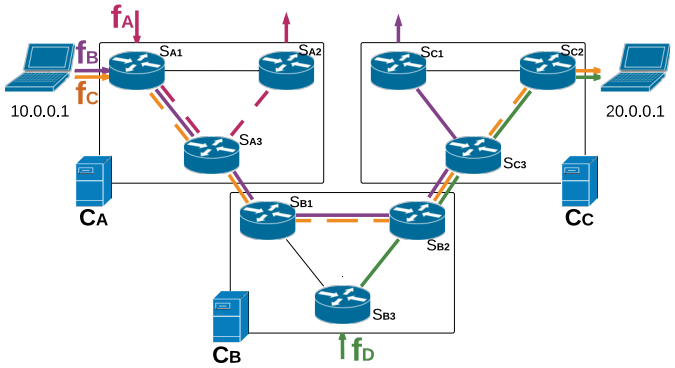


Fig. 2: Flow f_A traverses only the network of controller C_A, while flow f_B, f_C traverse all three networks and, thus, must be monitoring at all three controller. Flow f_D traverses the networks of C_B and C_C and is, therefore, measured redundantly in C_B and C_C [14].

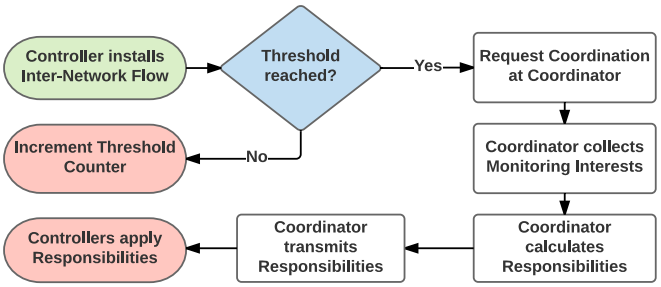


Fig. 3: Flow diagram showing the workflow of DISTTM: After a threshold of new inter-network flows a coordination is executed with the central coordinator in a focal position and leads to assignment of flow measurements without redundancy [14].

denoted DISCO. However, they limit to monitoring of inter-network links used for inter-controller communication instead of generally monitoring traffic or the data-plane as such. Levin et al. [22] discuss advantages and complexity when introducing physical distribution in control-planes.

Despite general collaboration between controllers for network management in distributed control-planes there is only little investigation with regard to monitoring in multi-controller and multi-network environments, respectively. For legacy networks, Terzis et al. [23] proposed collaboration for bandwidth brokers in different domains to allocate resources optimally.

Consensus protocols are distributed algorithms developed to agree on common decision in the context of a distributed system. Two prominent consensus protocols are PAXOS [24] and RAFT [25]. RAFT was developed to overcome the complexity of PAXOS, yet, solves the identical problem. It divides the consensus problem into three subproblems: Leader election, log replication and safety. PAXOS and RAFT could be used to agree on a leader and distribute state afterwards. However, the complexity and functionality both protocols provide exceeds our requirements for a per-measurement decision by far and would introduce unnecessary overhead. Nevertheless, the algorithm proposed in this work is inspired by RAFT.

III. Background: DistTM Distributed Traffic Matrix Generation

In this section we describe the fundamentals of the DISTTM [14] system that we extend in this work. Despite the fundamentals, we outline the limitations and justify the need of the work in hand.

A. *Fundamentals of* DistTM

As already broached in the introduction, redundant flow measurements introduce unnecessary costs in networks consisting of multiple sub-networks managed with a distributed control-plane. Flows that traverse multiple adjacent networks get measured in every network they traverse using naive, traditional approaches, e.g., to generate a traffic matrix [21]. To eliminate redundancy, we lately proposed DISTTM [14], a collaborative system for flow measurements with distributed control-planes.

The DISTTM system generates a per-controller traffic matrix for every managed network. The system fills the cells of the traffic matrix with the amount of traffic in bytes. This metric serves as an example and is exchangeable with other metrics that merely remain stable over the flow's path. Variations in the metric, e.g. due to congestion, must be captured using other techniques [26]. First, every controller updates the matrix using locally performed measurements. To this end, the DISTTM system mimics the behavior described in the OPENTM approach proposed by Tootoonchian et al. [21], thus, it communicates with the routing application to get informed on new flows. Occurring flows trigger a periodic measurement task for the respective flow with a configurable frequency. As already solved in OPENTM, we do not cover the switch selection in this work and use, for the sake of simplicity, a randomly selected switch on the flows path. Regarding the example depicted in Figure 2, controller C_A measures flow f_A, f_B, and f_C as all of them traverse its network. Furthermore, also controller C_B and C_C measure flow f_B and f_C since both traverse all networks. Flow f_D traverse networks of C_B and C_C, which naturally measure the flow. As a consequence, f_A is measured once, f_B, f_C three times, and f_D twice although the amount of traffic can be assumed merely stable. DISTTM's goal is to eliminate the redundant flow measurements, so that the total number of measurements per period reduces from nine to four in this example.

In its original design DISTTM intents one of the participating controllers as the coordinator. Figure 3 shows the behavior described in the following. Whenever a controller exceeds a certain number of new measurements, denoted coordination threshold, it requests a coordination at the coordinator. The coordinator itself gathers new measurement information from all participating controllers and calculates a fair distribution of responsibilities among the controllers. The assigned responsibilities are dispatched to each controller. In addition, this responsibility assignment messages include the information

Rhaban Hark, Khalil Tounsi, Amr Rizk, and Ralf Steinmetz. *Decentralized Collaborative Flow Monitoring in Distributed SDN Control-Planes (accepted for publication)* In: Proceedings of the Conference on Networked Systems (NetSys'19), p: 1–8, IEEE, March 2019.

about controllers that are also interested in the corresponding flow measurements. Subsequently, controllers measure only the flows they got assigned and receive measurements for other flows that traverse their network from other controllers of the federation. Given the example of Figure 2, the coordinator might assign f_B to C_A, F_C to C_B, and f_D to C_C. f_A is not included in the coordination process as it only traverses one subnetwork. After that procedure, every flow is measured only in one network. The load is balanced fairly among the controllers according to different strategies; and measurements are shared on the control-plane. If the number of flows increases, the costs for coordination and exchanged statistics on the control-planes become lower than the cost produced for redundant measurements as shown in the evaluation [14].

DISTTM included four basic exemplary fairness strategies: *(i) Fair Controller Distribution:* the number of measurement responsibilities is fairly distributed among controllers; *(ii) Fair Domain Distribution:* the number of measurement responsibilities is fairly distributed among domain assuming that multiple controllers can be part of different domains; *(iii) Fair Switch Distribution:* the number of measurement responsibilities is fairly distributed among controllers based on the number of switches in their network; algorithm *(iv) Random Distribution:* the measurement responsibilities are assigned randomly to one of the participating controllers. The evaluation show of DISTTM shows a proof-of-concept for the given fairness strategies.

B. Limitation: Relying on Central Coordination

As already mentioned earlier, the main limitation of DISTTM is the introduction of a central coordinator. The dependency on a single controller implies all vulnurabilities that usually occur with a single point of failure (SPOF): First, if the central coordinator breaks, the full concept of coordination cannot be applied further. In addition, in the case of network partitioning, only the parts of the network that are still connected to the coordinator can participate any longer. Despite this, although it might be negligible, there is additional load on the coordinating controller reducing the offload effect.

To solve this issue of having a central coordinator, in this work, we develop a distributed coordination algorithm that assigns measurement responsibilities among participating controllers robustly.

IV. DECENTRALIZED COLLABORATIVE TRAFFIC MATRIX ESTIMATION

In this section, we describe our approach to distribute monitoring responsibilities in multi-subnetwork SDNs without a centralized coordinator for the example of flow measurements to generate a per-controller traffic matrix. First, we briefly recap the initial isolated generation of traffic matrices, described in the next subsection, which is avoided by decentralized coordination. Last, we discuss our approach to fairly distribute the load on involved controllers.

A. Isolated Traffic Matrix Generation

The isolated flow monitoring works analog to the OPENTM [21] and, thus, the DISTTM [14] approach: The monitoring module interacts with the routing module that informs about flows arriving and vanishing from the network. For this, we define a flow as all packets processed with the same flow table entry. This definition is sufficient to capture all traffic within a network as long as the controller does not aggregate rules on a switch-level basis but a network-level basis. Once a new flow occurs, the traffic matrix generation application creates a new periodic measurement task and measures the traffic at a randomly selected switch on the flow's path. It uses the measurement results to update the table entry for the ingress-to-egress switch pair of the corresponding flow.

The procedure is always applied so that the controller generates the traffic matrix even if the coordination algorithm fails and as long as the coordination is not finished.

B. Decentralized Calculation and Assignment of Measurement Responsibilities

This subsection covers the main contribution of the work in hand. It describes the distributed algorithm performed to assign responsibilities for redundant measurements to single controller within the federation.

Whenever the routing application informs our monitoring about a new flow, first, the controller starts measuring the flow by itself as described in the previous subsection. Furthermore, the application launches the algorithm to find out whether the flow is measured redundantly and to solve this deficiency. Hence, for each flow one instance of the algorithm runs. We split the algorithm into three states that it may traverse: *(a) backoff*; *(b) piggybacked*; and *(c) measure* state. As Figure 4 indicates, the algorithm first visits the backoff state upon the arrival of a new flow and after starting periodic measurements for this flow.

a) Backoff state: In the backoff state, each controller generates a random backoff time within a defined period. After that, a controller naps for the given backoff before it enters the next state. If the backoff time is over without interruptions, the algorithm jumps into the measure state (cf. Figure 4).

However, if a controller receives an advertisement for the corresponding flow during the backoff time, the algorithm leaves the backoff time early and jumps in the piggybacked state.

b) Piggybacked state: By default, the controller does barely anything in the piggybacked state as it is "carried" from another controller - except informing the carrying controller about his interest beforehand (cf. *send_confirmation()* in Figure 4).

As a controller reaches a state through a received advertisement for a flow measurement from another controller, it always accepts the first offer. To do so, the controller sends a confirmation to the advertising controller including information about its interest in the measurements and its identity. After that, it stops its own measurements of the corresponding flow and the algorithm idles.

Rhaban Hark, Khalil Tounsi, Amr Rizk, and Ralf Steinmetz. *Decentralized Collaborative Flow Monitoring in Distributed SDN Control-Planes (accepted for publication)* In: Proceedings of the Conference on Networked Systems (NetSys'19), p: 1–8, IEEE, March 2019.

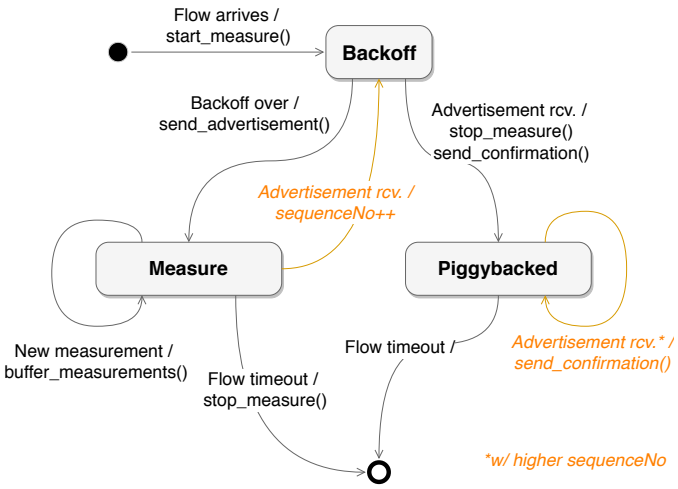


Fig. 4: State diagram showing the states of the algorithm: for a new flow the backoff state is reached; afterwards the algorithm goes to the measurement state or the piggybacked state depending on whether an measurement advertisement has been received or not.

In that state, the controller performs no measurements for this flow. However, as the controller that took over the measurement responsibility informs other interested controllers periodically on updates for the measurement, the traffic matrix gets updated implicitly using these statistic updates.

c) Measure state: If the backoff state was not interrupted with an advertisement, the algorithm reaches the measure state. In the measurement state, the controller assumes it is the first one to reach this state and takes over the responsibility to measure the corresponding flow. To this end, in advertises itself by dispatching an advertisement message to all other controllers including a unique flow identity, e.g. the 5-tuple, and its identity.

As each controller starts measuring each flow before the algorithm starts, there is no need for interaction regarding the measurements. Nevertheless, the measurements must be shared with interested controllers. In the measure state, the controller maintains a list of interested controllers per flow measurement, which is filled using the confirmation messages the other controllers dispatch when reaching the piggybacked state. Thus, after each measurement the controller checks if another controller is interested in the results and, if so, it inserts the value in a buffer available for every other controller as shown in Figure 4.

This controller flushes these buffers using the same period as it measures. Flushing a buffer means that it checks for every controller whether statistic updates are available. If there are statistics available, it sends them as batch to the interested controllers. Certainly, as multiple statistics are packed as batch in a single packet the costs to share measurements between controllers is lower than measuring each flow redundantly even for low numbers of flows.

C. Fairness Enforcement Efforts

The proposed algorithm includes a mechanism that tries to enforce fairness among controllers regarding their load. In the backoff state, the algorithm delays its offer (instantiated in an advertisement message) to measure the flow on behalf of all controllers. The generated backoff time allows to increase and decrease, respectively, the probability to be selected. Hence, we propose an exemplary mechanism to adjust this backoff time.

In order to estimate how much a controller is loaded compared to the other controllers, the algorithm considers the ratio of own measurement tasks and all measurement tasks. Therefore, each message dispatched to another controller contains the current number of a controller's measurement responsibilities. Subsequently, each controller knows the total number of measurements conducted from other controllers. First, we calculate a reference value b_{ref} for the backoff using

$$b_{ref} = F \cdot m_{assigned}^2 \cdot {m_{total}}^{-1}. \quad (1)$$

Larger numbers of assigned measurements ($m_{assigned}$) increase this value much more than smaller values for $m_{assigned}$ due to the square. Furthermore the value is set relative to the total number of measurements (m_{total}) and a configurable factor (F) is applied. By default we set $F = 3s$.

Furthermore, we pick a random number from a continuous uniform distribution between 0 and 1 and scale the reference backoff value by this factor. This is done to avoid having two controllers pick a close-by backoff. However, in early stages of the evaluation it turned out that the fairness was not satisfying as the influence of the randomly picked factor was higher than expected compared to the influence of the reference value. To solve this, we propose to rather pick a random backoff from a Gaussian distribution with $\mu = b_{ref}$ as mean and $\sigma^2 = 0.1 \cdot b_{ref}$ as variance. Thus:

$$b \sim \mathcal{N}(b_{ref}, 0.1 \cdot b_{ref}) \quad (2)$$

with b being the final backoff waiting time. Subsequently, the backoff time is randomly picked close to the reference value that considers the number of own measurements in comparison with the total number of measurements.

D. Collisions

As discussed lately controllers might pick backoff times that are close to each other, in particular, if the number of measurements is fairly distributed among them. It might happen that a controller C_A dispatches an advertisement message as its backoff time elapsed and before another controller C_B received the advertisement, C_B's backoff time elapsed as well and it dispatches another advertisement message. We call such a situation a collision.

To solve a collision, we introduce rounds of the algorithm identified by sequence numbers. The algorithm always starts using the sequence number 1. Once a controller detects a collision, which might happen at similar times in all controllers that calculated a close-by backoff time, the following procedure,

Rhaban Hark, Khalil Tounsi, Amr Rizk, and Ralf Steinmetz. *Decentralized Collaborative Flow Monitoring in Distributed SDN Control-Planes (accepted for publication)* In: Proceedings of the Conference on Networked Systems (NetSys'19), p: 1–8, IEEE, March 2019.

indicated with italic (orange) text in Figure 4 is performed. Each controller detecting a collision let the algorithm move back to the backoff state after increasing the sequence number for this flow. Back in the backoff phase, the algorithm behaves as already known from this phase and, thus, calculates a new backoff time. We do not increase the new backoff time as known from CSMA/CA that uses exponential backoffs, as the number of involved participants is limited. Again, the first controller to send an advertisement is responsible for the measurement and shares them with all controllers that confirm their interest. All messages for the flow containing a lower sequence number are ignored. Controllers, that have already confirmed their interest to a controller, subsequently having the algorithm in the piggybacked state, might confirm their interest to the new controller advertising for the flow if the advertisement contains a higher sequence number. Hence, once a controller is in the piggybacked state it never leaves it, but might confirm to another controller.

If a controller fails, other controllers do not receive further statistic updates. After two subsequently missing updates of a flow, the controllers go back into to backoff state after increasing the sequence number and re-elect the monitoring responsibility assignment.

V. Evaluation

This section covers the evaluation of the developed algorithm. We evaluate the system against the cost reduction and its fairness among the participating controllers. First, we briefly describe the evaluation environment including the developed prototype and the evaluation methodology. Afterwards, we show results of the conducted measurements.

A. Evaluation Environment

We evaluate the distributed monitoring approach using Mininet [27] as network emulator running on a simulation server (*Ubuntu 16.04 x64 server; 24 Intel(R) Xeon(R) cores @ 2.60 GHz; 128 GB Memory*). To control the network and performing the traffic matrix generation using our proposed approach, we use a Floodlight OpenFlow controller[1]. We implement the monitoring and all required functions as Floodlight modules. In addition to the traffic measurement module that fills the traffic matrix, we develop a module to execute the algorithm, hence, for the collaboration with other controllers. The module uses a developed communication module between all controllers using out-of-band channels. Furthermore, we add modules to detect hosts and peering-points with other network-parts as well as a module to perform shortest-path routing among multiple sub-networks based on a built-in topology detection module.

As topology for our evaluation, we use a distributed data-center scenario. In such as scenarios multiple data-centers belonging to the same domain are interconnected with each other. However, each data-center is controlled by a controller each. As topology, we followed [28] that use the following topology: In the highest layer, so-called *Fat-Cat* switches are interconnected as ring. In the next layer, multiple *cluster* switches are interconnected as ring again. Each cluster switch is connected to all Fat-Cat switches. A data-center has multiple clusters, whereas the clusters are not connected with each other except through the Fat-Cat layer. In each cluster, multiple racks are connected with a *rack* switch per rack. Again, each rack switch is connected to all cluster switches within the cluster. The cluster switch has a number of servers connected to it. In our evaluation, we use three small datacenters connected through their first Fat-Cat switch linearly. Each datacenter has two Fat-Cat switches, two clusters with two cluster switches each and two racks in each cluster. We substitute all servers in a rack into a mininet host.

We generate flows that randomly occur between randomly selected racks of different datacenters. The flows use small bandwidth normally distributed around a mean of 0.5 MBits (scale of 0.1 Mbits). Note, that the actual traffic load does not influence the metrics gathered in this evaluation and, thus, is kept low to avoid high load on the evaluation machine. The traffic is generated using iperf[2]. The racks produce flows with an exponentially distributed inter-arrival time with a scale of 1.0, thus, as Poisson process. We assume that scaling up the number of flows linearly influences the evaluation results. The flow duration is changed as independent parameter. The monitoring module measures each active flow with a frequency of $1/5s$, which is also used as statistic sharing frequency.

B. Evaluation Methodology

We repeat the evaluation so that we have for each metric at least 30 measurements, e.g. while measuring the costs per controller we conducted 10 runs with one data point per controller (3 controllers). Each run had a runtime of 4 minutes plus a 45 seconds startup phase.

We capture three metrics for the evaluation. First, the number of statistic requests requested per controller per evaluation run to show the elimination of redundant measurements. Furthermore, the total costs in terms of statistic requests, statistic replies, advertisement messages, confirmation messages, and statistic update messages between controllers show the total monitoring costs. Lastly, to evaluate the fairness we again use the ratio of measurements per controller relative to the total number of measurements conducted in the network.

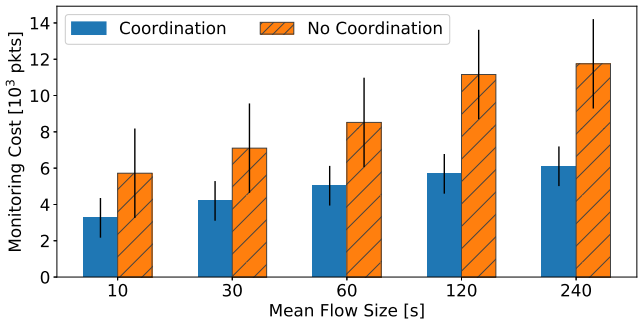
C. Evaluation Results

In the following, this subsection describes the results of the conducted evaluation to show *(i)* a proof-of-concept for the decentralized redundancy elimination algorithm, *(ii)* the relative cost reduction in our scenario compared to the naive non-cooperative approach, and *(iii)* the impact on the fairness among controllers.

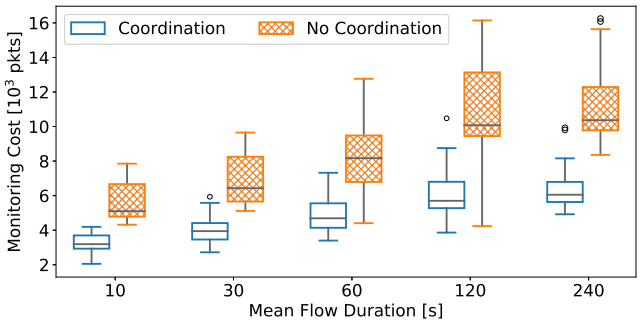
[1]Project Floodlight: Floodlight OpenFlow Controller http://www.projectfloodlight.org/floodlight/, [Access: Oct 25, 2018]

[2]iPerf - The ultimate speed test tool for TCP, UDP and SCTP https://iperf.fr/, [Access: Oct 25, 2018]

Rhaban Hark, Khalil Tounsi, Amr Rizk, and Ralf Steinmetz. *Decentralized Collaborative Flow Monitoring in Distributed SDN Control-Planes (accepted for publication)* In: Proceedings of the Conference on Networked Systems (NetSys'19), p: 1–8, IEEE, March 2019.



(a) Required statistic requests per controller per evaluation run while changing the mean flow duration.



(b) Costs for all monitoring related messages including the coordination process per controller per evaluation run while changing the mean flow duration.

Fig. 5: The evaluation depicted in Subfigure (a) shows a reduction of statistic requests performed from controllers when redundant measurements are eliminated using the proposed algorithm. Furthermore, Subfigure (b) shows a strong reduction of total costs required for the monitoring when the cooperative approach is used. Both figures indicate that longer flow durations increase the gap between the costs required for the cooperative and the naive approach.

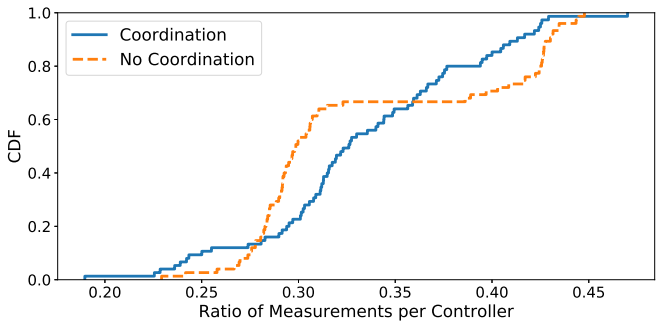


Fig. 6: Distribution of the ratio of measurements conducted per controller. The figure indicates that the fairness can be improved and controllers with focal positions can be relieved from unproportional load using the proposed algorithm.

1) Eliminate Redundancy through Decentralized Coordination: In Figure 5a, we find the number of statistic requests dispatched per controller per run on the y-axis. On the x-axis, we compare different mean flow durations on a logarithmic scale. The shown flow duration is the average in seconds of a normal distribution with scale of 5 seconds. The figure shows that the average number of dispatched statistic requests strongly decreases using the cooperative approach, shown in blue without hatches, compared to the naive approach, shown in orange with hatches. While we increase the average flow duration, we observe that the number of statistic requests increases for both the cooperative and the non-cooperative case.

This barplot shows that the system takes away a noticeable number of statistic requests from the overall monitoring. As flows are monitored redundantly as long as the coordination has not been finished and due to the occurrence of flows only between two networks the number does not decrease to a third of the naive solution.

D. Total Monitoring Cost Reduction

Figure 5b depicts the cost in terms of the number of messages used for all monitoring related communication per controller per run on the y-axis. The x-axis shows again the mean flow duration in seconds. As seen in the figure, the total cost to monitor the network traffic and set up the traffic matrix in each controller reduces strongly when coordination among controllers is used to eliminate redundant measurements. For each of the box combinations, the median number of messages for monitoring without coordination, shown in orange with hatches, is higher than the upper quartile when using the proposed coordination, shown in blue without hatches. Furthermore, we observe that the monitoring cost have a higher variance without coordination.

Looking at the median, we observe that the cost reduction increases by increasing the flow duration. On the one hand, considering short flows, the influence of the coordination algorithm is relatively higher in comparison to the statistic gathering cost reduction. On the other hand, for long flows the costs for coordination take a minor role compared to the cost reduction by eliminating redundant measurements. Hence, the approach improves when it measures long flows.

1) Fairness among Network-Parts: As the higher variance in the monitoring costs already indicates it, the fairness among the controllers cost is not optimal using a non-cooperative. Due to the focal position of the middling datacenter (remember that we had three datacenter linearly connected), it sees a higher number of flows compared to the outer datacenters as we randomly select flow endpoints from the datacenter racks. Figure 6 shows that such a scenario increases the load on the controllers that sees more flows than the other controllers do. The figure shows the distribution of the ratio of measurements per controller. As the orange line indicates, after approximately $^2/_3$ there is a shift in the distribution, meaning that one controller conducts a much higher number of measurements compared to the others. Using the cooperative

Rhaban Hark, Khalil Tounsi, Amr Rizk, and Ralf Steinmetz. *Decentralized Collaborative Flow Monitoring in Distributed SDN Control-Planes (accepted for publication)* In: Proceedings of the Conference on Networked Systems (NetSys'19), p: 1–8, IEEE, March 2019.

approach and its fairness attempt that takes the ratio of measurement responsibilities into account leads to a fairer distribution as indicated with the blue line. Although the number of measurements per controller is not perfectly equal we are able to eliminate the harsh shift.

In the evaluation we show that the number of statistic request and the total costs can be reduced when redundant measurements are eliminated using the proposed algorithm. Furthermore, we show that the fairness, namely the ratio of measurements conducted by each controller, can be improved with the cooperative approach.

VI. Conclusion

In this work, we develop a distributed algorithm that allows a decentralized coordination between controllers of a distributed-control plane to assign redundant measurement tasks to a single controller in order to remove the redundancy. Based on a prior system denoted DistTM that estimates a per controller traffic matrix while eliminating redundant measurements, we propose a sequel to this without the need for centralized controller. The proposed algorithm uses partly randomly generated backoff times to ensure load fairness among all controllers with respect to the number of assigned measurement tasks.

An evaluation study showed the proof-of-concept of monitoring cost reduction and an investigation of the flow duration showed that monitoring longer flows benefits more from the eliminated redundancy. Furthermore, we measured an improvement of the load fairness among controllers when using the proposed algorithm.

We left the development of further strategies to calculate the backoff time in order to achieve even better fairness to future work. Also, the algorithm benefits from further investigation regarding the optimal selection of system parameters such as the reference backoff time or the measurement-sharing interval.

Acknowledgment

This work has been performed in the framework of the CELTIC EUREKA project SENDATE-PLANETS (Project ID C2015/3-1), and it is partly funded by the German BMBF (Project ID 16KIS0471). The authors alone are responsible for the content of the paper. This work has been partially funded by the German Research Foundation (DFG) within the Collaborative Research Center (CRC) 1053 - MAKI as part of subprojects B1 and B4.

References

[1] Open Networking Fundation, "Software-Defined Networking: The New Norm for Networks," *ONF White Paper*, 2012.

[2] M. Yu, L. Jose, and R. Miao, "Software Defined Traffic Measurement with OpenSketch," in *Proc. USENIX NSDI*, 2013.

[3] Y. Yu, C. Qian, and X. Li, "Distributed and Collaborative Traffic Monitoring in Software Defined Networks," in *Proc. ACM SIGCOMM Workshop HotSDN*, 2014, pp. 85–90.

[4] S. Shirali-Shahreza and Y. Ganjali, "FleXam: Flexible Sampling Extension for Monitoring and Security Applications in Openflow," in *ACM SIGCOMM Workshop HotSDN*, 2013.

[5] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. Madhyastha, "FlowSense: Monitoring Network Utilization with Zero Measurement Cost," in *Passive and Active Measurement*, M. Roughan and R. Chang, Eds. Springer, 2013, vol. 7799, pp. 31–41.

[6] Y. Zhang, "An Adaptive Flow Counting Method for Anomaly Detection in SDN," in *Proc. ACM CoNEXT*, 2013.

[7] N. van Adrichem, C. Doerr, and F. Kuipers, "OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks," in *Proc. IEEE/IFIP NOMS*, 2014.

[8] D. Raumer, L. Schwaighofer, and G. Carle, "MonSamp: A Distributed SDN Application for QoS Monitoring," in *Proc. IEEE/ACM FedCSIS*, 2014, pp. 961–968.

[9] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards an Open, Distributed SDN OS," in *Proc. ACM SIGCOMM Workshop HotSDN*, 2014.

[10] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A Distributed Control Platform for Large-scale Production Networks," in *Proc. USENIX OSDI*, 2010.

[11] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controllers," in *Proc. IEEE/IFIP NOMS*, 2014.

[12] A. Tootoonchian and Y. Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow," in *Proc. USENIX Workshop INM/WREN*, 2010.

[13] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications," in *Proc. ACM SIGCOMM HotSDN*, 2012.

[14] R. Hark, D. Stingl, N. Richerzhagen, K. Nahrstedt, and R. Steinmetz, "DistTM: Collaborative Traffic Matrix Estimation in Distributed SDN Control Planes," in *Proc. IFIP Networking*, 2016, pp. 82–90.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[16] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming Protocol-independent Packet Processors," *SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[17] P. Tune and M. Roughan, "Internet Traffic Matrices: A Primer," in *ACM SIGCOMM eBook: Recent Advances in Networking*, H. Haddadi and O. Bonaventure, Eds., 2013, vol. 1, pp. 108–163.

[18] A. Medina, C. Fraleigh, N. Taft, S. Bhattacharyya, and C. Diot, "Taxonomy of IP traffic Matrices," in *Proc. SPIE ITCom*. International Society for Optics and Photonics, 2002.

[19] Y. Vardi, "Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data," *Journal of the American Statistical Association*, vol. 91, no. 433, pp. 365–377, 1996.

[20] Q. Zhao, Z. Ge, J. Wang, and J. Xu, "Robust Traffic Matrix Estimation with Imperfect Information: Making Use of Multiple Data Sources," *SIGMETRICS Performance Evaluation Review*, vol. 34, no. 1, pp. 133–144, 2006.

[21] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "OpenTM: Traffic Matrix Estimator for OpenFlow Networks," in *Passive and Active Measurement*, A. Krishnamurthy and B. Plattner, Eds. Springer, 2010, vol. 6032, pp. 201–210.

[22] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically Centralized?: State Distribution Trade-offs in Software Defined Networks," in *Proc. ACM SIGCOMM Workshop HotSDN*, 2012.

[23] A. Terzis, L. Wang, J. Ogawa, and L. Zhang, "A Two-Tier Resource Management Model for the Internet," in *Proc. IEEE GLOBECOM*, 1999.

[24] L. Lamport, "Paxos made simple," *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.

[25] D. Ongaro and J. Ousterhout, "In Search of an Understandable Consensus Algorithm," in *Proc. USENIX ATC*, 2014, pp. 305–320.

[26] R. Hark, N. Richerzhagen, B. Richerzhagen, A. Rizk, and R. Steinmetz, "Towards an Adaptive Selection of Loss Estimation Techniques in Software-defined Networks," in *Proc. IFIP Networking*, 2017, pp. 1–9.

[27] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-defined Networks," in *Proc. ACM HotNets*, 2010.

[28] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the Social Network's (Datacenter) Network," in *Proc. ACM SIGCOMM*, 2015, pp. 123–137.