DistTM: Collaborative Traffic Matrix Estimation in Distributed SDN Control Planes

Rhaban Hark*, Dominik Stingl*, Nils Richerzhagen*, Klara Nahrstedt‡ and Ralf Steinmetz*
*Multimedia Communications Engineering Lab, Technische Universität Darmstadt, Germany
‡Department of Computer Science, University of Illinois – Urbana, USA
*{rhaban.hark|dominik.stingl|nils.richerzhagen|ralf.steinmetz}@kom.tu-darmstadt.de ‡klara@illinois.edu

Abstract—**Recently, several works propose monitoring approaches for the emerged paradigm of Software-defined Networking. These provide a couple of ideas to retrieve various information about the network state leveraging new concepts for monitoring data collection at flow-level. As existing approaches reduce their scope to networks with a single controller, even sophisticated approaches ignore a potentially great efficiency gap, due to redundant flow measurements by multiple controllers in adjacent networks. To show a possibility how to close this efficiency gap, we propose a solution for collaborative traffic matrix estimation, termed DistTM. It exploits the property that flows traverse multiple networks and are monitored by several controllers. Through collaboration, the resulting monitoring tasks are coordinated and distributed among participating controllers to capture relevant information about all traversing flows, omitting redundant data collection. Conducted simulations reveal that DistTM operates efficiently: the monitoring traffic is significantly reduced, while the traffic matrix entry staleness is slightly affected. Furthermore, DistTM provides different schemes for a fair load balancing on controllers and switches while taking different influencing aspects into consideration.**

I. Introduction

Traffic monitoring constitutes the basis for nearly all network management functions. Since monitoring should always be non-invasive, thus, rather passive, yet robust, accurate and timely, a trade-off between performance and monitoring costs is omnipresent. This makes monitoring an inevitable challenging, however, required task. With the utilization of Software-defined Networking (SDN) [14], that provides a separation of the control and data plane, new possibilities evolve to measure traffic in the data plane. Using new techniques, the need for additional intelligence at forwarding elements and additional dedicated collection devices, as was required in traditional monitoring solutions, such as sFlow[1] or NetFlow [4], vanishes. Lately, several approaches propose a variety of monitoring tasks, leveraging these new techniques of SDN, in particular flow-level counters. These approaches measure, for instance, basic metrics, like link utilizations, delay, and packet loss [3], [21], [22]. Sophisticated approaches collect aggregated information (e.g. traffic matrices [19]) or execute more advanced monitoring tasks, such as heavy hitter detection [7].

So far, current approaches limit the control and measurement of networks to single controllers. However, to (i) avoid a single point of failure, (ii) provide scalability and (iii) relieve a controller from frequent polling of the switches, relying on single controllers is not recommended. As a consequence, physically distributed control planes are taken into account.

This work proposes a first step towards controller collaboration for monitoring in SDNs. The benefits of the controller collaboration are shown at the example of distributed traffic matrix estimation. Traffic matrices provide useful means for network provisioning, route planning, and further management tasks [20]. The proposed approach, called DistTM, makes use of the property that traffic flows most likely take paths through multiple adjacent network portions. The system hinders the networks to monitor the flows each. More precisely, it avoids the redundant measurement of flows traversing through multiple networks that are managed by different controllers. Instead, using DistTM, the controllers collaborate to coordinate monitoring tasks and share information in the control plane.

DistTM can be utilized inside single-administrated domains, such as data center networks (*intra-domain* collaboration), as well as, competing, adjacent domains (*inter-domain* collaboration) and is applicable in both scenarios. As discussed later, controllers must expose only a minimal amount of information to other controllers. In addition, domains are not forced to provide capacity which is not equally provided in return.

One further contribution of this work is a fair distribution of load among controllers and switches. Since the system determines traffic matrices collaboratively, we propose different load distribution schemes to assign responsibilities for flows among participating controllers or switches in a fair manner. As different scenarios require different schemes for load distribution, we show three elementary fairness schemes for different purposes and scenarios.

The conducted simulations of DistTM show, that the system significantly reduces the overall monitoring traffic overhead. Thus, it lowers the load on the controllers, whereas it affects the accuracy in terms of staleness of matrix entries only slightly. Furthermore, simulations show that we attain a fair distribution of the overhead for switches and controllers.

The remainder of the paper is structured as follows: Section II discusses the relevant background and related approaches. Section III describes the design of DistTM. Further on, Section IV presents the preliminary evaluation while Section V concludes the paper.

[1]sFlow Overview, http://www.sflow.org/about/ [Access: Oct 21, 2015]

ISBN 978-3-901882-83-8 © 2016 IFIP

II. Background and Related Work

This section gives an overview over helpful backgrounds and relevant works which contribute to this paper. The first subsection briefly introduces mechanisms to capture data plane information in SDNs, before the second subsection specifies traffic matrices of interest. Additionally, it gives a short introduction to OpenTM, a traffic matrix estimation approach for single-controller SDNs. Its basic concept of how a traffic matrix can be determined is used as reference for non-collaborating networks. Subsequently, the third subsection describes selected works about collaboration between network nodes. To the best of the authors' knowledge, collaborative monitoring for SDNs on controller level, as it is targeted in this work, has not yet been investigated.

A. Data Capturing in Software-defined Networks

In OpenFlow [12], the widely accepted de-facto standard for Ethernet-based SDNs, the data plane consists of dump switches with flow tables managed by controllers in the control plane. In addition to a number of management fields, these flow tables comprise counter fields which are incremented every time a packet is processed using the corresponding entry. For every entry a packet as well as a byte counter are available for monitoring use. These can be deactivated individually to reduce the load on a switch. The controller is supposed to fetch counters in multiple ways: (i) using explicit statistic requests for single flow entries or aggregations; or (ii) implicitly when a flow entry is removed (e.g. triggered by a timeout). The OpenFlow switch specification version 1.5[2] introduces additional thresholds for counters, providing push-based counter access. A related OpenFlow extension that is denoted FleXam [16] provides additional packet sampling capabilities. In contrast, approaches such as OpenSketch [23] and DCM [24] implement a specialized data plane which make it again necessary to customize the protocol, yet yielding to efficient data collection mechanisms.

Anyway, most existing monitoring approaches, rely on version OpenFlow protocol version 1.3 which is expected to be a stable basis[3]. In order to reach applicability in non-custom SDN environments, this work only uses features available in the original protocol.

B. Traffic Matrices of interest

Traffic matrices are abstract data structures showing traffic information for pairs of network nodes. As an example and also used in this work, a traffic matrix can accumulate the amount of traffic in terms of packets or bytes between all ingress/egress switch pairs in the network. They are used for management tasks like capacity planning, network provisioning, load balancing policies for route optimization, but can also be used to detect traffic anomalies and other security related events [20]. They might contain different representations, such as a maximum, minimum, average or a sum. Besides traffic matrices, other types of matrices, like delay matrices, presenting the node-to-node delay for all pairs, or loss matrices exist. Further studies as well as a taxonomy can be found in [13], [20].

Tootoonchian et al. propose OpenTM [19] to estimate traffic matrices in the context of SDN. It takes advantage of the logical centralization of the controller. More precisely, it uses centrally available information given by the controllers routing application to observe upcoming flows. Further on, it queries a flows path in order to be able to select one switch on the path and poll it for statistics. By accumulating flow level byte counters of flows originating at the same node and ending at the same node, it calculates all entries of the traffic matrix. Aside traffic matrix estimation, OpenTMs main contribution is an intelligent selection of polled switches on the path in order to support a fair overhead distribution among switches and yet perform well in terms of accuracy due to packet loss. However, this work adopts the matrix estimation concept of OpenTM based on available routing information.

C. Collaboration in SDN

In the context of network collaboration, Yu et al. [24] propose a memory efficient collaboration-enabled control plane for SDNs. The work states that flows are often monitored redundantly at different switches if flow aggregation is used to reduce the number of rules. On the other side, if single flows are selected to reach fine granular measurements, the number of rules becomes too large. They tackle the problem using two-stage Bloom filters on switches. These filters can be defined in a way the switches monitor particular sets of flows without the need to define one rule per flow. Thus, as rules can be defined efficiently in alignment with monitoring rules of other switches, DCM allows collaboration on switch level. In contrast, we try to achieve collaboration on the controller level. Terzis et al. [17] already proposed a model for collaboration on a comparable level in 1999 in another context. In their approach, bandwidth brokers of different domains maintain agreements to cooperatively allocate resources for *inter-domain* traffic.

In order to be able to collaborate with other controllers, recent approaches introduce infrastructures for distributed SDN control planes [2], [5], [9], [15], [18]. Their target is to give controllers of the same control plane a shared view on the whole network while distributed properties are abstracted as good as possible for their applications. Those approaches do not include monitoring as a potential controller application for distributed control planes. However, DISCO [15] introduces monitoring agents in controllers which are able to measure link utilization. Actually, as the monitoring is limited to links between peering points to adjacent networks and each controller measures the statistics individually, no collaboration is done in this context. In this work it is assumed that the control plane allows controllers to communicate with one another. Hence, a distributed control plane infrastructure, such as DISCO,

[2]ONF: SDN Resources – Technical Library, https://www.opennetworking.org/sdn-resources/technical-library [Access: Oct 22,2015]

[3]Sean Michael Kerner: OpenFlow Protocol 1.3.0 Approved, http://www.enterprisenetworkingplanet.com/nethub/openflow-protocol-1.3.0-approved.html [Access: Oct 26, 2015]

would be a good basis to satisfy the need for a communication possibility between controllers. Levin et al. [11] point out, that trade-offs between staleness and optimality as well as between application complexity and robustness turn up when logically centralized control planes are mapped to physically decentralized.

III. DistTM

This section deals with the concept of the distributed collaborative traffic matrix estimation system, termed DistTM. DistTM consists of collaborating modules that are installed at multiple controllers and exchange messages among each other for the estimation of traffic matrices at the controllers. Therefore, a controller periodically interacts with its switches to capture the data, as detailed in Section III-A. Given this interaction pattern, the collaboration and coordination among controllers for the distributed estimation of traffic matrices are presented in Section III-B and Section III-C, respectively. Finally, Section III-D introduces three schemes to influence DistTM's coordination for a fair task distribution based on different criteria.

A. *Generation of Traffic Matrices*

The problem of estimating a traffic matrix in an SDN with a single controller can be tackled using, for instance, OpenTM [19] as described in Section II-B. A controller informs its traffic matrix estimation module whenever a new flow is installed. Afterwards, the statistics for this flow are periodically polled from a selected switch on the flow's path. The interval for the periodic polling is specified by the system parameter T (*polling request interval*). To select a switch along a flow's path, OpenTM presents multiple strategies for an intelligent and sophisticated switch selection. However, DistTM relies on a random selection of a switch along the path for the sake of simplicity and directs to OpenTM for questions relating the switch selection. The traffic matrix module in a controller uses the gathered statistics from the selected switch to generate the traffic matrix and update the affected cell of the matrix. For the identification of the correct cell, the controller uses the origin (ingress switch) and the destination (egress switch) of the flow. Given the example in Figure 1, C_A polls statistics of flows f_A, f_B and f_C. Regarding f_A, the statistics can be polled at switch S_{A1}, S_{A3} or S_{A2} along its path. The measurements are used to fill the cell of pair $(S_{A1}, S_{A2}) = (ingress, egress)$. C_B polls statistics of f_B and f_C to fill the cell of pair (S_{B1}, S_{B2}) as well as f_D for (S_{B3}, S_{B2}). C_C acts analogously in its network. A flow is polled until the controller receives a message about its timeout (*FlowRemoved*). In this work, DistTM stores the total amount of bytes for each cell. Other metrics, such as the throughput or bandwidth consumption, may be calculated or derived as well.

The described interaction between a controller and its switches represents an easy-to-operate solution. However, the simplicity of this approach comes at the expense of inevitable excessive cost in terms of statistic requests at switches and

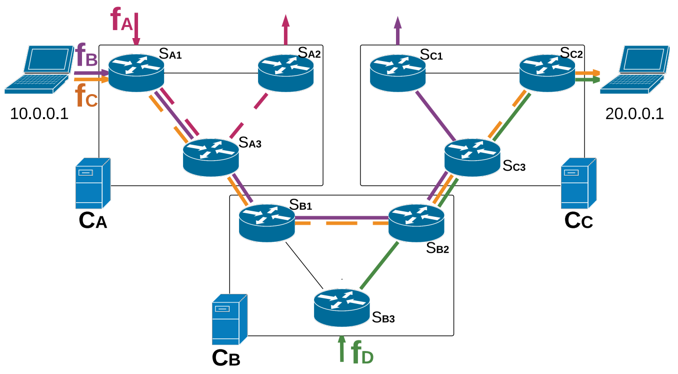


Fig. 1. Inter-network flows.

controllers, particularly questioning its scalability. To reduce the overhead and improve scalability, DistTM breaks up with the necessity to measure each upcoming flow on each controller. Instead, we introduce the concept of controller collaboration, i.e. sharing of information, in the following.

B. DistTM *Controller Collaboration*

Assuming the knowledge about peering points between adjacent networks, flows which traverse through networks of cooperating controllers can be identified. As an example, C_A detects, that f_B and f_C exit its network towards the network of C_B. Given this capability and knowledge about traversing flows, DistTM provides the functionality to coordinate the monitoring. Hence, controllers are able to trigger a coordination whenever considered necessary to distribute responsibilities for redundantly measured flows like f_B and f_C among the controllers. A coordination assigns each flow to exactly one controller, which is subsequently exclusively responsible to capture statistics of this flow (e.g. only C_A captures f_B and C_B captures f_C). So, DistTM informs each controller (i) about the flows it must monitor and (ii) about other controllers that are interested in the collected statistics of the corresponding flows. As a result of that coordination, in addition to periodic polling of statistics, controllers must transmit updates to a list of interested controllers. These actions are periodically triggered depending on the system parameter T.

Given the example in Figure 1, flow f_B originates at host `10.0.0.1` and has its destination at host `20.0.0.1`, traversing the network of controller C_A, C_B, and C_C. In the naive solution controller C_A, C_B, and C_C would poll statistics of flow f_B from selected switches in their network to update their traffic matrix with the resulting traffic of f_B. Using DistTM, only one controller, for instance C_A, would fetch the statistics of f_B and provide the information to C_B and C_C. In turn, other flows, such as f_C and f_D, may be assigned to C_B and C_C, while the resulting measurements are transmitted to all interested controllers. As flow f_D only traverses the network of C_B and C_C, C_A is even not aware of f_D and will not be informed about updates of f_D. Flow f_A is only monitored by C_A as it only traverses the network of C_A.

For this particular example the number of statistic requests reduces from 9 ($C_A : [f_A, f_B, f_C] + C_B : [f_B, f_C, f_D] + C_C : [f_B, f_C, f_D] = 9$) to 4 ($C_A : [f_A, f_B] + C_B : [f_C] + C_C : [f_D] = 4$).

A controller uses directly polled values and received values from other controllers to update its traffic matrix. In the example C_A uses the polled values of f_A to update the matrix entry for pair (S_{A1}, S_{A2}). To update the (S_{A1}, S_{A3}) entry it uses the directly polled values of f_B and complements it with the received values from C_B about f_C.

Among the collaborating controllers DISTTM identifies redundantly monitored flows and controllers interested in these flows. Based on these insights DISTTM selects one controller out of the pool of interested controllers to assign the flow to. The chosen controller is subsequently responsible to poll statistic values about the flow and share them with all interested controllers. Hence, only a fraction of all traversing flows must be monitored per controller. Besides the reduction of monitored flows, the controller collaboration also influences the number of transmitted messages as well as of transmitted bytes. Exchanged information about multiple flows between two controllers can be batched into one so-called *statistic batch message*, whereas direct polling of values from multiple flows must be performed per flow. Furthermore, a controller proactively transmits information to interested controllers at the end of each polling interval. This proactive transmission avoids unnecessary transmissions to request the statistics.

So far, DISTTM basically focuses on a distributed and fair procedure for collaborative data collection. As a result, it currently relies on static per-flow polling intervals as well as a static information exchange intervals between controllers. For future work, improvements concerning the polling of values by a controller in its own network, such as flow aggregation [25] or adaptive polling [3], are applicable to minimize the traffic or improve the performance.

C. DISTTM *Coordination*

As described, the system shares responsibilities for flows among collaborating controllers. For this task DISTTM delegates the coordination to one controller which is denoted as *coordinator*. In this work, a static configuration is used to select the coordinator. Nevertheless, leader election algorithms from other research areas (e.g. [1]) are applicable that deal with this problem and allow an autonomous and distributed election of a coordinator. Using a central coordinator simplifies the distribution of flow responsibilities among the controllers, as described hereafter.

1) Coordination trigger: The advantage of DISTTM comes at the expense of coordination overhead. This overhead composes of coordination requests, responsibility assignments and statistic exchange messages. Although the evaluation will show that the impact of additional messages is reasonable, it still needs to be considered. In order to avoid too many coordinations and limit the resulting coordination overhead in networks and scenarios with strong flow fluctuations, DISTTM introduces a counter in every participating controller

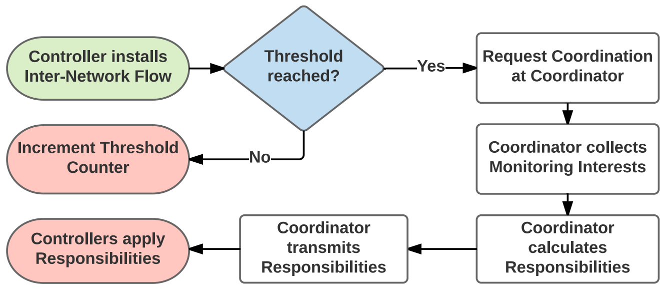

Controller installs Inter-Network Flow
Threshold reached?
Yes
Request Coordination at Coordinator
No
Increment Threshold Counter
Coordinator collects Monitoring Interests
Coordinator calculates Responsibilities
Coordinator transmits Responsibilities
Controllers apply Responsibilities

Fig. 2. Coordination mechanism flow diagram.

and a system parameter, referred to as *coordination threshold* R. The threshold may be set dynamically by the controllers based on other system parameters according to findings in the evaluation. At each controller, this counter is incremented every time a controller becomes aware of a new flow. For this, a controller takes only flows into account which have their destination in it's network but do not originate inside of the network. Based on this scheme, simultaneous coordination requests from multiple controllers are avoided since only the last network considers an upcoming flow. In the example given in Figure 1, assuming f_B, f_C and f_D leave the union of cooperating networks at the network controlled by C_C, the threshold counter of C_C would be set to three. If the incremented counter reaches the specified coordination threshold R a coordination procedure is triggered. The coordination procedure comprises the transmission of a so-called *coordination request message* from the controller to the coordinator. The counters are reset every time a coordination procedure is performed.

2) Coordination procedure: As described above, the coordinator executes a coordination procedure whenever it receives a coordination requests. As depicted in Figure 2, if the coordinator receives a coordination request message it collects the monitoring interests of all participating controllers. Therefore, it sends a *monitoring interest request message* to all participating controllers, asking for flows which occurred since the last coordination and traverse their network. The requested information is sent back to the coordinator, using a *monitoring interest response message*. Subsequently, the coordinator applies a *responsibility calculation function* based on the received interests. This responsibility calculation function is responsible to assign flows to the controllers. Furthermore, a list of interested controllers per flow is appended to inform the responsible controller about other interested controllers. The flow assignments including the list of interested controllers are transmitted to the corresponding controllers, using a *coordination instruction message*.

Concerning the example of Figure 1, the coordinator takes the information that controller C_A is interested in f_A, f_B and f_C; C_B and C_C are interested in f_B, f_C and f_D as depicted as input in Figure 3. The function may distribute the responsibilities, for instance, as follows: C_A is responsible to monitor its *intra-network* flow f_A. Furthermore, C_A is responsible to measure f_B and inform C_B and C_C about its

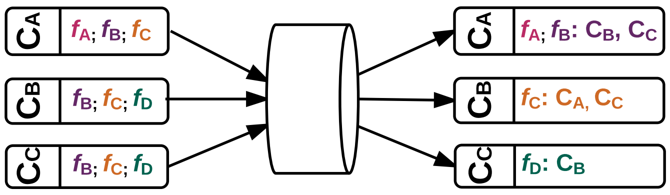

C_A | f_A; f_B; f_C
C_B | f_B; f_C; f_D
C_C | f_B; f_C; f_D
C_A | f_A; f_B: C_B, C_C
C_B | f_C: C_A, C_C
C_C | f_D: C_B

Fig. 3. Coordination function input and output.

updates. C_B needs to monitor f_C and inform C_A and C_C, while C_C is responsible for flow f_D and to inform C_B about measurement updates. In this case, excluding *intra-network* flows, the coordinator assigns one flow to each controller. Hence, it shares the responsibilities fair among controllers.

D. Fairness schemes

In addition to the relevant flows of a controller, a monitoring interest response message comprises further information. This covers the number of *intra-network* flows inside of a single network (e.g., f_A in C_A), the number of previously assigned, still active flows and the number of switches in the controllers network. The additional information is optional and may not be included due to privacy constraints. However, if it is included, the coordinator uses this information for the responsibility calculation function in order to apply various fairness schemes that influence the assignment of flows. DISTTM offers four fairness strategies that are applicable to distribute the responsibilities among controllers.

*1) Fair Controller Distribution **(FCD)**:* The *FCD* scheme distributes the responsibilities equally among all participating controllers in terms of monitored flows in total. Hence, this scheme makes sure that every controller is supposed to make the same number of statistic requests to switches in its network. This includes requests for *intra-network* flows not traversing networks controlled by other controllers. However, *FCD* leads to equal statistic request load among controllers and might be preferred, for instance, by single-administrated networks.

*2) Fair Domain Distribution **(FDD)**:* As different controllers may be part of competing domains, they are not willed to respect *intra-domain* flows of other networks. As a simplified scheme to support this, the *FDD* scheme distributes the responsibilities equally among controllers while *intra-network* flows are not taken into account. This scheme leads to an equal distribution of flows to monitor for other controllers. A more elaborated version of this scheme could consider fairness between each controller pair. A controller C_A, for instance, does neither want to respect a flow only in the network of C_B, nor traversing only the networks of C_B and C_C.

*3) Fair Switch Distribution **(FSD)**:* So far, the issue that networks may differ in size has been ignored. Consider two collaborating networks: one consisting of only two switches, while the other consists of an order of magnitude more switches. If flows traverse both networks with the same probability, a distribution based on the statistic requests a controller has to poll from a switch would lead to a disproportional load on the two switches of the small network. To tackle this issue, the *FSD* scheme assigns the responsibilities in a fair manner based on the current requests per switch. This leads to a fair load distribution among switches in the network, however, potentially not among controllers.

*4) Random **(RD)**:* As reference model, a random distribution among controllers is used. As it can be expected, for large experiments, the scheme resembles the fair controller distribution assuming equal flow occurrences per network.

The listed schemes optimize particular scenarios and situations. Hence, we propose to use dynamic combinations based on the characteristics and requirements in real-world scenarios. As a flow is assigned to a controller DISTTM does not dictate which switch needs to be selected on a flow's path to fetch statistics from. Consequently, the controllers are responsible to make a sensible selection. As already mentioned in Section III-A, related approaches tackle this aspect and is not taken into consideration in this work. However, it may be an important aspect for future work.

IV. EVALUATION

This section describes the evaluation of DISTTM. The major objectives of the evaluation are to show (i) that the system is able to reduce monitoring costs, while the performance can be maintained and (ii) to highlight the influence of the different fairness schemes of the responsibility calculation function on DISTTM. To be able to quantify this, we implemented DISTTM as a Floodlight[4] OpenFlow controller application as which it is deployed on all controllers in a virtual simulation network. In addition, we included a simplified routing application based on a modest host detection extension, the ability to detect peering points with other networks, and a straightforward westbound interface. The westbound interface enables the communication between controllers through TCP connections using an out-of-band controller network in the control plane. Using this small set of functionality, the distributed traffic matrix estimation system – DISTTM – is implemented.

A. Evaluation Methodology and Scenario

We use Mininet [10] as evaluation testbed to generate virtual OpenFlow-enabled networks. As sketched in Figure 4, for this preliminary evaluation, we choose a synthetic, linear topology with three individually controlled networks with three switches each as default scenario. The first network, controlled by C_A has a peering-point to the network controlled by C_B, which itself has a peering point at the other end to the network controlled by C_C. This topology allows a direct presentation and measurement of the DISTTM principles.

For the evaluation, we use the parameters shown in Table I. The first to two parameters represent DISTTM's system parameters, as introduced in the previous section. The following parameters except the last one configure the modeled flows in the network and are detailed in the following paragraph.

[4]Project Floodlight: Floodlight OpenFlow Controller http://www.projectfloodlight.org/floodlight/, [Access: Nov 09, 2015]

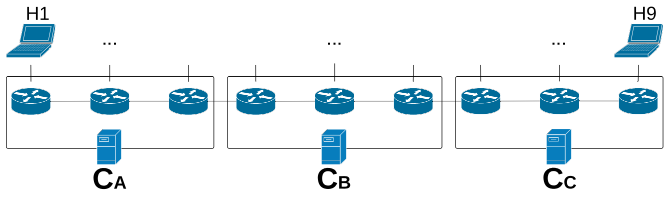


Fig. 4. Synthetic evaluation topology.

The last parameter F_{max} represents the flow idle timeout. If multiple values are listed for a parameter, the values for a default configuration are underlined.

TABLE I
SCENARIO AND SIMULATION SETUP.

Parameter	Values	Description
T [ms]	500, 1000, <u>2000</u>, 5000	Polling period
R	1, <u>2</u>, 3, 5, 10, 20	Coordination threshold
r [pps]	100	Packet rate
λ [s]	1	E[Flow inter-arrival time]
k	5	Pareto shape index
x_{min} [pkts]	500	Min. packets per flow
F_{max} [s]	5	Flow idle timeout

As only the duration of active flows instead of their size is relevant for the conducted evaluation, flows have a constant packet rate r and minimal packet size. For the inter arrival time of flows in seconds, we use an exponentially distributed stochastic variable with $\lambda = 1$ [8]. Concerning the length of a flow, a Pareto distributed stochastic variable is used. As parameters, $k = 5$ and $x_{min} = 500$ are set. Thus, most flows are small (slightly larger than 500 packets) and only a few are significantly large. A flow consisting of more than 500 packets is alive for at least 5 seconds with the specified packet rate. Assuming one second inter arrival time, if the system is in equilibrium, at least (and most commonly) six flows intersect at a time. We set the probability of a flow's source (H_i) and destination (H_j) being in one of the networks to $\frac{1}{3}$ each, with $i \neq j$. All simulations were repeated 50 times. Bar plots report the mean with 95% confidence intervals. Box plots report the median, lower and upper quartiles as well as whiskers for the fifth and 95th percentiles. The simulations were executed on a dedicated simulation machine[5].

During the evaluation, we investigated three different metrics to quantify DISTTM's behavior. At first, we measure monitoring costs of DISTTM using the number of statistic requests per controller per flow. This metric is independent of the evaluation duration as the number of flows increases linear with time if the system is in equilibrium. The second metric quantifies the overhead which is additionally produced by DISTTM. This overhead comprises the different types of coordination messages as well as statistic batch messages, which are exchanged between controllers. Similar to the first metric, the second one is measured per controller per flow. To enable a reasonable aggregation of the two metrics, in some cases the metric's presentation is in bytes instead of the number of messages or requests. Although one cannot compare statistic requests and coordination messages directly due to the fact that they do not share the same medium and lead to different load on the controllers, this compromise allows finding a comprehensible operating point for the trade-off between cost reduction and introduced overhead. Note, that the statistic requests with their replies are always of the same size, so that the qualitative behavior is similar for message counts as well as bytes. The resulting bytes of coordination messages may vary depending on the type and amount of shared information. To compare fairness, we selected the portion of statistic requests a controller or switch is supposed to trigger or answer, respectively, as metric. This is identical to the portion of flows a controller must monitor. In addition to these metrics that particularly quantify the costs of DISTTM, it must be shown, that the performance for the generation of the traffic matrices does not suffer using DISTTM. As packet losses are not considered in the scope of this evaluation, the performance is measured in terms of staleness of matrix entries. The staleness is defined by the period between two consecutive updates of a matrix entry for active flows.

[5] *Ubuntu 14.04.3 LTS / 24x2.6GHz Intel(R) Xeon(R) CPU / 128GB RAM*

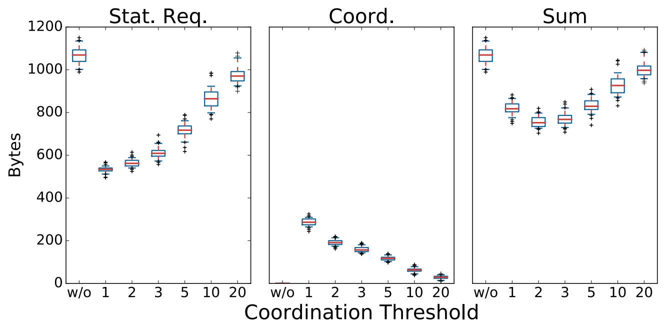


Fig. 5. Cost reduction evaluation.

B. Monitoring Cost Reduction

Figure 5 shows the main contribution of DISTTM . For the default polling period of $T = 2000\,ms$, the leftmost plot of Figure 5 shows the number of bytes used for the statistic requests and replies for one controller per flow. The number reaches its peak when DISTTM is not used (w/o). In that case, the non-collaborative approach is applied and every controller must monitor all flows it is interested in. Using DISTTM , the plot shows a significant reduction of the number of bytes for statistic requests. However, it becomes apparent that the traffic grows if the coordination threshold R is increased. This results from the fact that low coordination thresholds lead to frequent re-configurations and fast assignments of *inter-network* flows to single controllers. Consequently, redundant measurements occur fewer. For large thresholds, the occurrences of redundant measurements of flows increase, as each controller monitors all flows before they are monitored only once. Thus, as visible

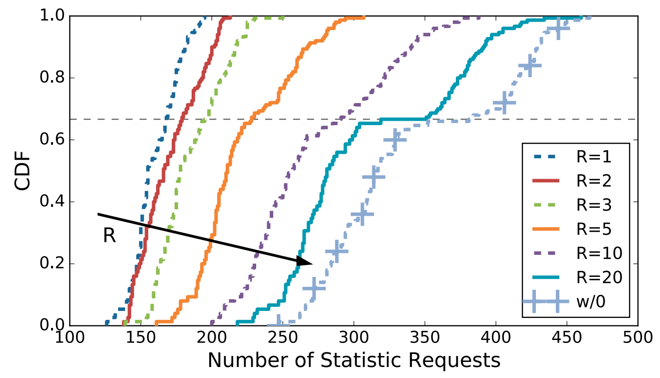


Fig. 6. Statistic request distribution among controllers.

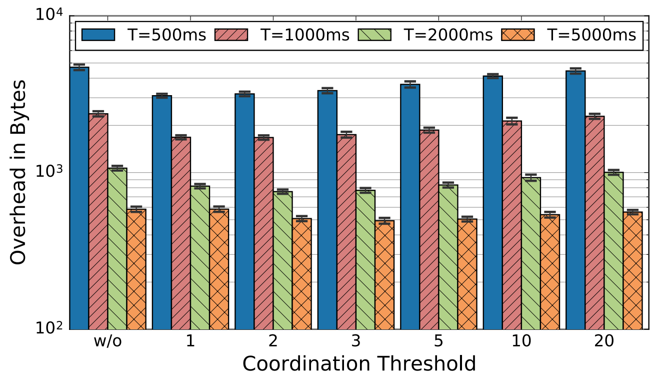


Fig. 7. Resulting traffic of the statistic requests and coordination control messages.

in the figure, the traffic from the transmitted statistic requests and replies increases as well. The plot in the middle of Figure 5 depicts the number of bytes needed for the different coordination messages. Without the use of DistTM this value is zero since no coordination between the controllers occurs. For a small threshold R this overhead is larger, while larger thresholds, thus, rarer coordinations, lead to less coordination control messages. The resulting coordination traffic in bytes decreases with increasing threshold. Finally, the rightmost plot of Figure 5 shows the total load in terms of bytes comprising the statistic requests and replies plus the number of bytes for coordination messages. As depicted, a trade-off between the coordination threshold R and the total load exists. Coordinating on every arising flow ($R = 1$), leads to less monitoring costs (flows are always monitored only by one controller instead of each), whereas the coordination overhead is relatively high. Increasing the threshold R leads to larger monitoring costs, whereas the coordination overhead decreases. For the evaluated scenario with $T = 2000\,ms$, the operating point is between $R = 2$ and $R = 3$.

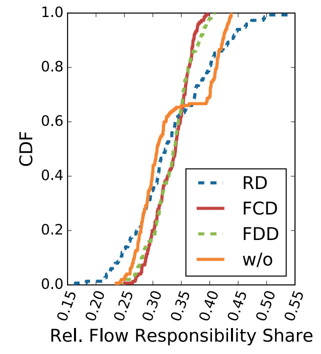
Figure 6 depicts the distribution of statistic requests per controller per flow for the different coordination threshold (R) configurations. It can be seen that the total number of requests is significantly lower using DistTM. For $R = 1$ the system must request the least statistics. Even for high thresholds, the number is still lower than for the classical approach without DistTM. The reduced number of statistic requests for a decreasing coordination threshold R originates from the fact the frequency of redundant, thus, unnecessary requests is reduced. As already mentioned, a lower R leads to more frequent re-configurations and fast assignment of redundantly measured flows to single controllers. In addition to the total amount of requests, the cumulative distribution function (CDF) depicts the distribution of the requests among the controllers. Regarding the distribution without the use of DistTM, we observe that the number of statistic requests has a shift after $\frac{2}{3}$ of the data points. In the given evaluation topology (cf. Figure 4) controller C_B must handle more flows in average than C_A and C_C, as it located in between. Due to this focal position and an equal distribution of flow sources and destinations, it must monitor more flows than the other two controllers. As a result, an unfair distribution occurs. DistTM balances the distribution more if responsibility for flows are shared between the controllers. For large thresholds, the distribution is still less balanced due to the larger number of redundantly monitored flows. With a lower coordination threshold, the negative impact of C_B's focal position nearly disappears due to frequent re-configurations and the faster unique assignment of flows to controllers. A more elaborated view on fairness and load balancing is presented in Section IV-D.

C. *Impact of the Polling Period*

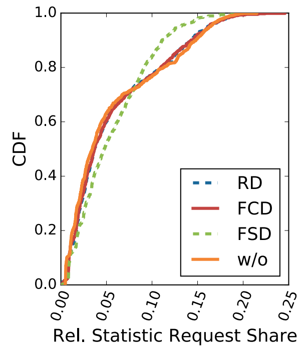
Figure 7 depicts the overall traffic (note the logarithmic scale) comprising the statistic requests and replies plus the coordination control messages for a varying statistic polling period T and a varying coordination threshold R. As expectable, it can be observed that an increasing polling period reduces the overall traffic. However, for shorter polling periods the total savings are higher than for longer periods, whereas relative savings are comparable, although hardly identifiable in the figure. Apart from that, it is observable that the polling period influences the operating point. For a short period, e.g., $T = 500ms$, the lowest total overhead is at $R = 1$. As the period is extended, e.g., $T = 5000ms$, the operating point shifts to larger thresholds ($R \approx 3$). Hence, for short polling periods, the coordination overhead has less influence on the total load. The other way around, lower frequencies lead to larger influence of coordination control messages, thus minimizing the positive effect of DistTM and the collaborative estimation of traffic matrices.

D. DistTM *Fairness*

In order to examine the fairness, Figure 8a depicts the share of flows a controller has to monitor in relation to the total number of flows monitored by all controllers. For the use of DistTM, three fairness schemes are applied and compared to the classical approach (w/o DistTM). The classical approach and the *RD* scheme show rather unfair distributions. For the classical approach, C_B suffers from its central position in the topology, as shown by the shift for $\frac{1}{3}$ of all measurements. In contrast, the *FCD* scheme and the *FDD* scheme (cf.



(a) Flow responsibility distribution among controllers.



(b) Statistic request distribution among switches.

Fig. 8. Fairness evaluation results

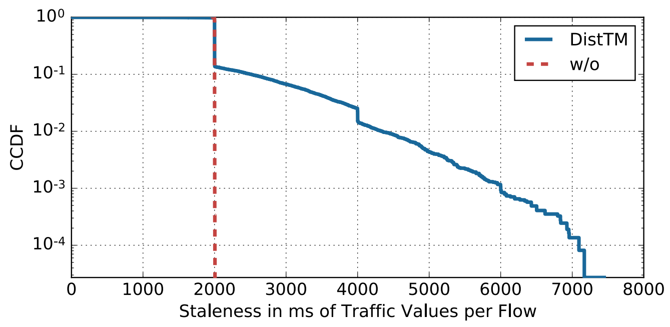


Fig. 9. Traffic matrix entry staleness.

Section III-D) lead to a steep curve indicating a balanced distribution of responsibilities among the controllers. Thus, the proposed schemes improve the load fairness on controllers. Due to the fact that *intra-network* flows occur in all three networks with the same probability, *FDD* and *FCD* behave similarly.

For the next experiment, the network of controller C_C is enlarged and consists of four times more switches. However, the probability for a flow starting or ending in one network is unchanged. Hence, the switches in the small networks may be polled more often in average when *FCD* is applied. Figure 8b depicts the portion of statistic requests in relation to all triggered requests that must be processed by a switch. It becomes apparent that the fairness in terms of load on a switch is improved using the *FSD* scheme. This results from the fact that the *FSD* scheme is the only scheme, which considers the current state of the switches.

$$f(x_1, .., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2} \tag{1}$$

In order to capture the fairness by a single value, we calculate the *fairness index (FI)* proposed by Jain [6]. Equation (1) produces a single value between 0 and 1, where 1 reports the highest fairness.

TABLE II
OBSERVED CONTROLLER AND SWITCH FAIRNESS.

Scheme	Controller Fairness	Switch Fairness[6]
RD	0.9510	0.5377
FCD	0.9893	0.5400
FDD	0.9907	——
FSD	——	0.6736
w/o DISTTM	0.9677	0.5413

[6]Adapted topology: the network of C_C contains four times more switches; equal probability of flow source and destination per network.

Using the unmodified scenario, the *FCD* and *FDD* schemes balance the load among the controllers in the fairest manner. Both FIs are qualitatively equal, as listed in Table II and already depicted in Figure 8a. For the modified scenario with the adapted topology the second column of Table II outlines the fairest load balance among the switches if the *FSD* scheme is applied. A fair balancing of statistic requests between the switches cannot be provided if fairness schemes are applied that ignore the different number of switches in the network as well as the current load on switches.

E. Impact of DISTTM *on Performance*

To examine potential trade-offs, it must be analyzed to which extent the traffic matrix estimation performance is influenced. We use the matrix entry staleness to represent the performance. For the sake of simplicity, all controllers poll statistics with the same frequency. In the classical approach, the delay between the actual values and the available data in the controller consists only of the statistic reply transmission delay between the switch and the controller. Using DISTTM, this delay is extended by two further delays. At first, the delay between a received statistic reply and the end of the measurement period, where a statistic batch message is exchanged between two controllers is added. Furthermore, the transmission delay of this statistic batch message between the controllers must be added. Since the statistic polling interval is typically much larger than the transmission delay, they can be neglected. Figure 9 depicts the CCDF of the traffic matrix entry staleness for the default statistic polling rate of $T = 2000\,ms$. It becomes apparent that about 90% of the values are updated every 2000 ms (note the logarithmic scale). However, some values are staler using DISTTM. Whenever a coordination is triggered and its instructions are sent to a controller, the statistics of a flow which is assigned to another controller are not polled anymore. The next update for this flow arrives with the next statistic batch message of the responsible controller. In the meantime the affected entries are not updated. As the figure shows for a polling period of $T = 2000\,ms$, most remaining entries are updated in less than $2 \cdot 2000ms = 4000ms$. Shorter intervals are also possible during the coordination procedure. Altogether, DISTTM leads to an exponentially distributed excess staleness.

After a coordination the staleness is restored to T.

F. Summary

The results of the evaluation demonstrate that DISTTM significantly reduces monitoring costs resulting from a reduced number of required statistic requests per controller. However, introduced coordination messages lead to a trade-off between monitoring costs and additionally added overhead. We control this trade-off through a coordination threshold which controls the frequency of coordinations and improves the flow assignment. Furthermore, the results reveal that the staleness of matrix entries remains the same for approximately 90% of all measurements. Apart from that, the evaluation shows that the introduced fairness schemes allow an adjustable and fairer load distribution among controllers and switches, respectively.

V. CONCLUSIONS

In this paper we show how the collaboration between SDN controllers significantly reduces monitoring overhead while sacrificing a negligible performance fraction. The key to this reduction is a collaborative system in a distributed SDN control plane denoted DISTTM that divides the monitoring tasks such that redundant flow monitoring is eliminated. We use DISTTM to exemplarily collect estimates of traffic matrix information. We measure the performance of the monitoring system using the traffic matrix entry staleness. The empirical results show an exponentially distributed excess staleness when DISTTM is utilized. We explore the assignment of monitored flows to corresponding controllers based on different criteria. For example, we show the impact of different fairness allocations on the respective controller and switch load distributions. Note that the SDN controllers need not to be in the same domain. We keep the investigation using real world network traces and topologies for future work. Further investigations will also explore adaptive polling rates per controller and dynamic fairness schemes.

ACKNOWLEDGMENT

This work has been funded in parts by the German Research Foundation (DFG) as part of project B1, C2 and C3 within the Collaborative Research Center (CRC) 1053 – MAKI. Parts of this work have been conducted within the SENDATE project, a project funded by the German Federal Ministry of Education and Research (BMBF). The authors would like to thank Julius Rückert and Amr Rizk for their valuable input and contributions.

REFERENCES

[1] B. Awerbuch, "Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election, and Related Problems," in *ACM Annual Symposium on the Theory of Computing (STOC)*, 1987.

[2] 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 *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2014.

[3] S. Chowdhury, M. Bari, R. Ahmed, and R. Boutaba, "PayLess: A low Cost Network Monitoring Framework for Software Defined Networks," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2014.

[4] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954, October 2004.

[5] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2012.

[6] R. Jain, *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, 1991.

[7] L. Jose, M. Yu, and J. Rexford, "Online Measurement of Large Traffic Aggregates on Commodity Switches," in *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, 2011, pp. 1–13.

[8] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido, "A nonstationary Poisson view of Internet traffic," in *IEEE International Conference on Computer Communications (INFOCOM)*, vol. 3, 2004, pp. 1558–1569.

[9] 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 *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2010.

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

[11] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically Centralized?: State Distribution Trade-offs in Software Defined Networks," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2012.

[12] 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.

[13] A. Medina, C. Fraleigh, N. Taft, S. Bhattacharyya, and C. Diot, "Taxonomy of IP traffic Matrices," in *SPIE ITCom: The Convergence of Information Technologies and Communications.* International Society for Optics and Photonics, 2002.

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

[15] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controllers," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2014.

[16] S. Shirali-Shahreza and Y. Ganjali, "FleXam: Flexible Sampling Extension for Monitoring and Security Applications in Openflow," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2013.

[17] A. Terzis, L. Wang, J. Ogawa, and L. Zhang, "A Two-Tier Resource Management Model for the Internet," in *IEEE Global Communications Conference (GLOBECOM)*, 1999.

[18] A. Tootoonchian and Y. Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow," in *USENIX Internet Network Management Workshop/Workshop on Research on Enterprise Networking (INM/WREN)*, 2010.

[19] 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.

[20] 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.

[21] N. van Adrichem, C. Doerr, and F. Kuipers, "OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2014.

[22] 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.

[23] M. Yu, L. Jose, and R. Miao, "Software Defined Traffic Measurement with OpenSketch," in *USENIX Symposium on Network Systems Design and Implementation (NSDI)*, 2013.

[24] Y. Yu, C. Qian, and X. Li, "Distributed and Collaborative Traffic Monitoring in Software Defined Networks," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2014.

[25] Y. Zhang, "An Adaptive Flow Counting Method for Anomaly Detection in SDN," in *Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2013.