REMO: Resource Efficient Distributed Network Monitoring

Tao Li[1], Hani Salah[2], Mu He[3], Thorsten Strufe[1], and Silvia Santini[4]

[1]TU Dresden, Germany
[2]Palestine Polytechnic University, Palestine
[3]TU Munich, Germany
[4]Universitá della Svizzera Italiana (USI), Switzerland
{*tao.li,thorsten.strufe*}@tu-dresden.de, hani@ppu.edu, mu.he@tum.de, silvia.santini@usi.ch

***Abstract*—Increasing the traffic visibility, by monitoring network flow packets, provides valuable information for various network management tasks. The mirroring mode of flow packet monitoring requires the switches and routers to duplicate packets of interest, and to send them to flow monitors for in-depth analysis. A common practice to avoid the interference between the original and the mirrored flows is to transmit them separately, in two different planes (data plane and monitoring plane, respectively). In this paper, we aim at reducing the overall cost of transmitting both the original and mirrored flows. Towards that end, we present a generic monitoring framework called REMO. The key idea of REMO is twofold: (i) placing the flow monitors in central locations, and (ii) passing the original flows through the vicinity of the monitors. By doing so, REMO reduces the resources consumed in the monitoring plane, without unworthily increasing the resource consumption in the data plane. The results of extensive numerical simulations show that REMO effectively reduces the overall transmission cost, remarkably outperforming several baseline strategies, particularly when the transmission is more expensive in the monitoring plane.**

I. INTRODUCTION

Flow packet monitoring is an important network management method in present communication systems. In this method, packets are examined to provide valuable insights on data flows. For instance, it helps the network manager to diagnose network performance issues [1] or to identify hostile traffic [2], [3].

Network management and security services, such as intrusion detection systems and application-level fingerprinting, require the thorough examination on network packets [4], [5]. Software defined networking (SDN) based flow monitoring systems proposed in [6] and [7] allow the switches or routers to duplicate *original data flows* on the per-packet basis, according to the dynamically configured matching fields of data flows. The *mirrored data flows* are sent to flow monitors for analyses by predefined algorithms. The separation between the original data flows and the mirrored data flows (i.e. between the network's *data plane* and *monitoring plane*) is a common practice in network monitoring [8], [9]. It brings benefits to the overall performance and reliability of the monitoring system. For example, the congestion in the data plane will not affect the transmission of mirrored data flows in the monitoring plane, and vice versa [10].

The aforementioned separation can be achieved either physically or virtually. Original and mirrored data flows in the physical separation approach traverse through isolated networks built upon different underlying communication technologies. For example, in hybrid data center networks consisting of both 60 Ghz wireless links and wired optical fiber links [11], [12], the wireless links and the wired links can be dedicated to the mirrored data flows and to the original data flows, respectively. Due to the distinct energy consumption profiles of these two communication technologies, the resources required to transmit the original data flows and those required to transmit the mirrored data flows are different [13], [14]. Meanwhile, the virtual approach relies on the state-of-the-art network slicing technology [15], [16]. This technology allows to allocate exclusive slices of network resources for the original data flows and mirrored data flows. Depending on the QoS requirements of the allocated network slices, techniques like Dynamic Voltage and Frequency Scaling (DVFS) can be used to change the execution rates of the Network Processor Unit (NPU) allocated to the network slice [14]. In this case, it also costs different amounts of resources to transmit original and mirrored data flows in the distinct communication planes.

In this paper, we present REMO, a **R**esource **E**fficient distributed **MO**nitoring system. REMO is a general framework that optimizes the global resource consumption incurred by the transmission of original and mirrored data flows in isolated communication planes. REMO achieves this goal by employing a two-step consecutive scheduling approach: Firstly, REMO places the flow monitors at central places in the network, so that the mirrored data flows do not need to traverse long paths from the switch to the flow monitor. Secondly, in order to further reduce resource consumption, REMO leverages the data path programmability provided by SDN. It carefully selects the paths (either one of the shortest paths or a stretched one) for the original data flows to make them even closer to the flow monitors.

In practice, we use an Integer Linear Programming (ILP) model to optimize the flow monitor placement, and a Mixed Integer Linear Programming (MILP) model to optimize paths to embed original flows and the selection of switches to duplicate flows. More precisely, the result of the flow monitor placement decides to which switches a server running an instance of the flow monitor is attached. The result of flow embedding chooses the paths that original data flows should traverse. As for the result of mirroring switch selection, it determines which switches should duplicate flows. Furthermore, to overcome the complexity of directly solving the formulated MILP problem, we propose a heuristic algorithm to efficiently solve the problem.

We evaluate the performance of REMO (i.e. its ability to reduce the costs of transmitting original and mirrored data flows) through extensive numerical simulations. Furthermore, we compare REMO with several baseline strategies. Overall, the results show that REMO is very useful in reducing the overall communication cost. Compared with random strategies, REMO is able to save over 50% of the network resource consumption, when transmission in the monitoring plane is more expensive. Our results also show that increasing the number of flow monitors can reduce at least 10% of resource consumption compared with the case where only one flow monitor exists.

The remainder of this paper is organized as follows: we review the related work in Section II. Then, we describe REMO in Section III and evaluate it in Section IV. Lastly, we conclude the paper and suggest directions for future work in Section V.

II. Related work

Network monitoring assisted by the SDN technology has been investigated during the past years. According to the OpenFlow specification 1.3 [1], entries in the flow tables of the SDN switches are able to provide SDN controllers with flow statistics including their size and alive time, through querying and replying mechanisms. To perform more detailed analyses on data flows, the authors of [6] and [4] proposed to extend the OpenFlow protocol and software switch implementations to enable per-flow packet sampling. Resource efficient monitoring, however, is not among their considerations.

To save the resource consumption incurred by network monitoring, DISTM [17] and [18] reduce the number of flow statistic queries and replies by avoiding duplicated measurements on multiple switches for the same flow. CeMon [19] and [20] use the traffic engineering approach to reduce the resource consumption incurred by flow monitoring. These two approaches force the original data flows to intersect as much as possible, so that the controller can poll the flow statistics from the same switch at one time. They are able to reduce the resources consumed by the operation of network monitoring. However, the additional network resources caused by stretching the paths of original data flows are ignored. In [21], the authors investigated the problem of deploying virtualized flow monitors by dynamically adjusting the number of the used flow monitors and data paths based on the flow information.

[1]https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf

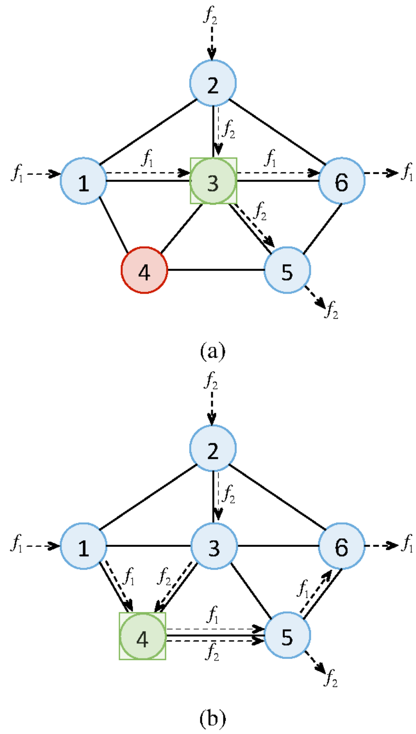


Fig. 1: Illustrative examples: (a) f_1 and f_2 are embedded to the links using the shortest path algorithm. The flow monitor is attached to Switch 3. (b) The flow monitor is attached to Switch 4. f_1 and f_2 are embedded to the links so as to force them to go through Switch 4.

Our work considers the total cost to transmit both original and mirrored data in different communication planes. The deployment cost of the flow monitors also can be integrated into our model. However, this is out of the scope of this paper.

III. System Model

We consider a distributed flow monitoring system relying on switches that can duplicate packets of selected data flows. There are several flow monitors deployed in the network and attached to the switches. Each switch is associated with one of the flow monitors. That is, a switch transmits all of its mirrored flow packets over the shortest path in the isolated monitoring plane to the same flow monitor. Besides, we assume that a (logically) centralized network monitoring coordinator continuously collects link cost information for both the monitoring and data planes. It computes the placement of flow monitors, and places the flow monitor instances on the desired locations in the network. The coordinator also obtains the global knowledge about the flows under monitoring, before or during the transmission of flows. The first case happens when a coflow exists. Coflow is a collection of parallel flows among hosts, occurring during the intermediate stages of big data applications [22]. The flow information can be known in advance by the network coordinator, through its channel with a data migration scheduler [23]. The second case happens when the network coordinator is notified that the flow bandwidth requirements change. Then the coordinator is responsible for calculating the paths for original data flows, as well as selecting the switches to duplicate them.

A. Illustrative examples

Our goal is to reduce the overall consumption of network resources used for transmitting both original and mirrored data flows. As mentioned in the introduction, we assume that transmission of mirrored data flows (in the monitoring plane) and transmission of original data flows (in the data plane) possibly are associated different costs.

We use toy examples (Figure 1) to motivate our work as well as to illustrate the problem which we aim to solve. We assume that the cost of transmitting a packet over one hop is 1 unit in the data plane and 5 units in the monitoring plane.

In Figure 1a, the flows f_1 and f_2 have the ingress/egress switch pairs $< 1, 6 >$ and $< 2, 5 >$, respectively. They follow the shortest paths. In this scenario, the minimum cost of network resource consumption (of 4 units including 4 units in the data plane, and 0 unit in the monitoring plane) is achieved by attaching the flow monitor to Switch 3, and also mirroring the two flows on Switch 3. Note that attaching the flow monitor to Switch 4, while keeping flow mirroring on Switch 3, increases the cost to 14 units (4 units in the data plane, and 10 units in the monitoring plane). In Figure 1b, we assume that the flow monitor is attached to Switch 4. If the flows f_1 and f_2 do not follow the shortest path anymore, but are embedded to the paths that contain Switch 4, the overall network resource consumption can be reduced from 14 units to 6 units (6 units in the data plane, and 0 unit in the monitoring plane).

To conclude, the overall network resource consumption is affected by the placement of flow monitors, the routing paths and flows mirroring locations. The monitoring coordinator is responsible for making three decisions: (i) where to place the flow monitors, (ii) which path the original data flow should traverse and (iii) which switch(es) should mirror data flows and later transmit the mirrored flows to the corresponding monitors.

B. Model description

In this subsection, we introduce our scheduling model that optimally reduces the overall network resource consumption. The model consists of two consecutive steps: (i) *flow monitor placement and switch binding (FMPSB)* and (ii) *flow embedding and mirroring switch selection (FEMSS)*. FMPSB depends on the topology information of the underlying network, and it takes place when the network topology evolves or the number of flow monitors changes. FEMSS requires the flow monitor placement results and the basic information about a batch of flows, including their ingress/egress switches and current bandwidth consumption.

Indeed, it is also possible to jointly compute FMPSB and FEMSS at one time and post the flow monitor placement and flow scheduling after the flow information is obtained [24]. However, we argue that it is preferable to compute flow monitor placement without prior knowledge about the data flows, to avoid frequent migration of the flow monitors.

The network topology we consider can be represented as a graph $G = (V, E)$, where V is the set of switches being capable of mirroring flow packets, and E is the set of links connecting them. We use k to denote the number of flow monitors that need to be placed, and $m \in V$ to represent a potential switch to attach a flow monitor. TABLE I lists the definitions of the variables used in the model.

Variable	Definition
V	set of switches in the network topology
E	set of links in the network topology
$D_{u,m}$	the distance of the shortest path from switch u to switch m
R^m, R^d	the cost to transmit data flows over one hop in the monitoring plane and in the data plane, respectively
$i :< s_i, d_i, b_i >$	flow info containing ingress switch s_i, egress switch d_i and the bandwidth consumption b_i
c_m	binary, whether to connect a controller to switch m
$a_{u,m}$	binary, whether switch u is associated with a flow monitor connected to switch m
$w_i^{u,v}$	binary, whether to embed a flow i into the link u, v
t_i^u	binary, whether a flow is mirrored on the switch u
$p_i^{u,v}$	continuous, voltage value on the link u, v for the flow i
q_i^u	binary, whether the flow i passes the switch u
B	the bandwidth capacity of a link
L_i	the shortest path length computed for the flow i

TABLE I: Variables used in the model

We detail the models to solve FMPSB and FEMSS in Subsection III-C and Subsection III-D, respectively.

C. Flow monitor placement and switch binding

The main goal here is to achieve the lowest average distance between switches and monitors in the monitoring plane, without the knowledge of the flow pattern. This is similar to our previously proposed object in [25].

$$\text{min. } \frac{1}{|V|} \sum_{u \in V, m \in V} a_{u,m} \cdot D_{u,m} \tag{1}$$

Monitor placement constraints:

$$\sum_{m \in V} c_m = k \tag{2}$$

$$\sum_{u \in V} a_{u,m} \leq |V| \cdot c_m, \text{ for } m \in V \tag{3}$$

$$\sum_{m \in V} a_{u,m} = 1, \text{ for } u \in V \tag{4}$$

Constraint (2) indicates that the number of flow monitors that needs to be placed equals to k. Constraint (3) states that switches use only one switch as the destination for mirrored flows if that switch is attached with a flow monitor. Constraint (4) implies that a switch has only one binded flow monitor and the mirrored data flows from the same switch are sent to that particular binded flow monitor.

D. *Flow embedding and mirroring switch selection*

Here, we optimize the paths to embed original data flows as well as the selection of mirroring switches. The objective is to minimize the sum of the resource consumption of transmitting mirrored data flows in the monitoring plane (represented by the first item), and that of transmitting original data flows in the data plane (represented by the second item). Note that the mirrored data flows in the monitoring plane follow the shortest path from the switch to the flow monitor.

$$\min\ R^m \sum_u \sum_m \sum_i t_i^u \cdot c_m \cdot D_{u,m} \cdot b_i + R^d \sum_{(u,v)} \sum_i w_i^{u,v} \cdot b_i \tag{5}$$

Flow conservation constraints:

$$\sum_{v \in neighbor(u)} w_i^{u,v} - \sum_{v \in neighbor(u)} w_i^{v,u} = \begin{cases} 1, u = s_i \\ -1, u = d_i \\ 0, otherwise \end{cases} \tag{6}$$

Constraint (6) forces the standard flow conservation rule in which a flow must leave a non-ingress/egress switch, if that flow passes through it.

Simple path constraints:

$$p_i^{u,v} \leq w_i^{u,v}, \text{ for all } i \tag{7}$$

$$\alpha q_i^u \leq \begin{cases} 1 + \sum_{(u,v)} (p_i^{u,v} - p_i^{v,u}) & u = d_i \\ \sum_{(u,v)} (p_i^{u,v} - p_i^{v,u}) & otherwise \end{cases} \tag{8}$$

$$q_i^u \geq (w_i^{u,v} + w_i^{v,u})/2 \tag{9}$$

The flow conservation constraint is not sufficient to remove disjoint paths, as shown in Figure 2a. Therefore, we use the voltage value based method proposed in [20] to remove the disjoint paths. In that method, each link traversed by a flow has a continuous voltage value $p_i^{u,v}$ indicated by Constraint (7). In Constraint (8), α is a small value (less than $1/|E|$), and the voltage value of the outgoing link must be larger than that of the incoming link.

$$\sum_{(u,v) \in E} w_i^{u,v} \leq 1, \text{ for } u \neq d_i \tag{10}$$

The generated non-simple path containing the cycles, as shown in Figure 2b, is a valid solution in [20]. However, the flow traffic still has to traverse attached cyclic links, which leads to additional bandwidth consumption in the data plane. Thus, we use Constraint (10) to further remove attached cyclic links and generate only simple paths as shown in Figure 2c.

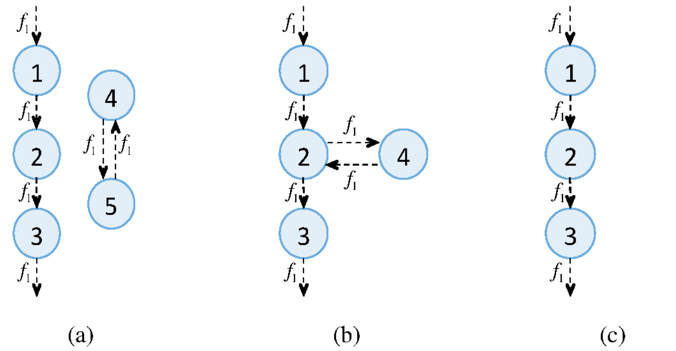


Fig. 2: Steps to generate a simple path containing no loops: (a) Flow path with disjoint paths; (b) Non-simple path with cycles after using the method proposed in [20]; (c) Simple path after removing the attached cycles.

Mirroring switch constraints:

$$t_i^u \leq q_i^u, \text{ for all } i \tag{11}$$

$$\sum_u t_i^u = 1, \text{ for all } i \tag{12}$$

Constraint (11) and Constraint (12) imply that only one switch belonging to the path of the original data flow can duplicate its packets.

Data plane bandwidth and path length constraints:

To avoid congestion in the data plane, Constraint (13) indicates that the aggregated bandwidth consumption on a data plane link should not exceed its capacity. Recall that we can stretch the paths of original data flows to decrease the overall resource consumption, when it costs more to transmit mirrored data flows in the monitoring plane than in the data plane. However, stretching the paths of original data flows means that the original data flow paths can be longer than their shortest paths. Thus, instead of aggressively pushing original data flows closer to the flow monitors, we take their path length into consideration and enforce an upper limit on it using Constraint (14). In this constraint, β is the path length stretch ratio, which is defined, for a flow i, as the ratio of the maximum allowed path length to the shortest path length L_i.

$$\sum_{(u,v) \in E} b_i w_i^{u,v} \leq B \tag{13}$$

$$\sum_{(u,v) \in E} w_i^{u,v} \leq \beta L_i, \text{ for all } (u,v) \tag{14}$$

The formulated MILP for flow embedding and mirroring switch selection is NP-complete. Thus, solving it using an optimizer requires a considerable amount of time when the problem size increases. To cope with this problem, we develop a heuristic algorithm (Subsection III-E) that provides a near-optimal performance for the formulated MILP and drastically reduces computation time.

E. Near-optimal heuristic algorithm

Algorithm 1 REMO-Heuristic: Main Procedure

Input: a batch of monitored flows F containing $f_i :< s_i, r_i, v_i >$, the flow monitor placement results
Output: the paths for original data flows and the switches to mirror them

Main procedure:
1: $A = \phi$
2: **for** f_i in F **do**
3: $\quad R = \phi$
4: $\quad L_{f_i}^{max} = \beta L_{f_i}^{shortest}$
5: $\quad$**for** each m_j in M **do**
6: $\quad\quad$Using [26], find all the shortest paths $\{p_{f_i}^1, p_{f_i}^2, ...\}$ with length n_j, from $ingress(f_i)$ to $switch(m_j)$
7: $\quad\quad$**for** index in $n_j, n_j - 1, .., 0$ **do**
8: $\quad\quad\quad$**for** each path p in $\{p_{f_i}^1, p_{f_i}^2, ...\}$ **do**
9: $\quad\quad\quad\quad switch = p[index]$
10: $\quad\quad\quad\quad$**if** $switch$ is not bound with m_j **then**
11: $\quad\quad\quad\quad\quad$continue
12: $\quad\quad\quad\quad$**end if**
13: $\quad\quad\quad\quad (s, r) = SearchPath(f_i, switch, p, L_f^{max})$
14: $\quad\quad\quad\quad$**if** $(s, r) == None$ **then**
15: $\quad\quad\quad\quad\quad$continue
16: $\quad\quad\quad\quad$**else**
17: $\quad\quad\quad\quad\quad R = R \cup (f_i, s, r)$
18: $\quad\quad\quad\quad$**end if**
19: $\quad\quad\quad$**end for**
20: $\quad\quad\quad$index=index-1
21: $\quad\quad$**end for**
22: $\quad$**end for**
23: $\quad$From R, pick up $(f_i, s\prime, r\prime)$ leading to the minimum network resource consumption
24: $\quad A = A \cup (f_i, s\prime, r\prime)$
25: **end for**
26: **return** A

We find experimentally that solving the current formulated ILP for the FMPSB subproblem only costs a small amount of time, even for a large-sized network. On the contrary, directly solving the MILP formulation for the FEMSS subproblem is a more complex task, which is unsuitable for large-sized networks with many flows. Thus, we design a heuristic algorithm, inspired by the deflection technique proposed in [27], to embed original data flows and select the switches to mirror their packets for the FEMSS subproblem.

The proposed near-optimal heuristic algorithm utilizes the shortest paths among the network switches. The main idea is to iterate through the switches on the shortest paths from the flow ingress switch to the flow monitors, and use the shortest path from the iterated switches to the flow egress switch as the remaining path for a original data flow. Then, the procedure chooses the switch and its associated path, that incurs the minimum communication resource consumption, as its mirroring switch and the original data flow path correspondingly.

Algorithm 2 $SearchPath(f_i, switch, p, L_f^{max})$

if no link on path p is congested **then**
$\quad$find all the shortest paths $\{q_{f_i}^1, q_{f_i}^2, ...\}$ from $switch$ to $egress(f_i)$
$\quad$**for** each path q in $\{q_{f_i}^1, q_{f_i}^2, ...\}$ **do**
$\quad\quad$**if** no link on path q is congested **and** $r = p \cup q$ is a simple path as in Figure 2c **and** $|r| \leq L_f^{max}$ **then**
$\quad\quad\quad$**return** $(switch, r)$
$\quad\quad$**end if**
$\quad$**end for**
end if
return None

As shown in Algorithm 1, the procedure firstly finds out the shortest paths from the ingress switch of a flow to all the switches attached with the flow monitors (line: 6). Then, the procedure inspects each switch on the identified shortest paths to check whether it can be used to duplicate the flow packets. The inspection starts from the closest switches to the flow monitors (lines: 7 – 22). If one switch is not associated with the current flow monitor, the algorithm skips to the next one (lines: 10 – 12).

The $SearchPath$ function, as shown in Algorithm 2, checks the status of the current path from the ingress switch to the potential mirroring switch, and rejects it if the path is congested. Afterwards, the algorithm starts searching the shortest path from the potential mirroring switch to the flow egress switch. A path gets accepted if it can form a complete data flow path without loops, and at the same time does not exceed the maximum path length constraint.

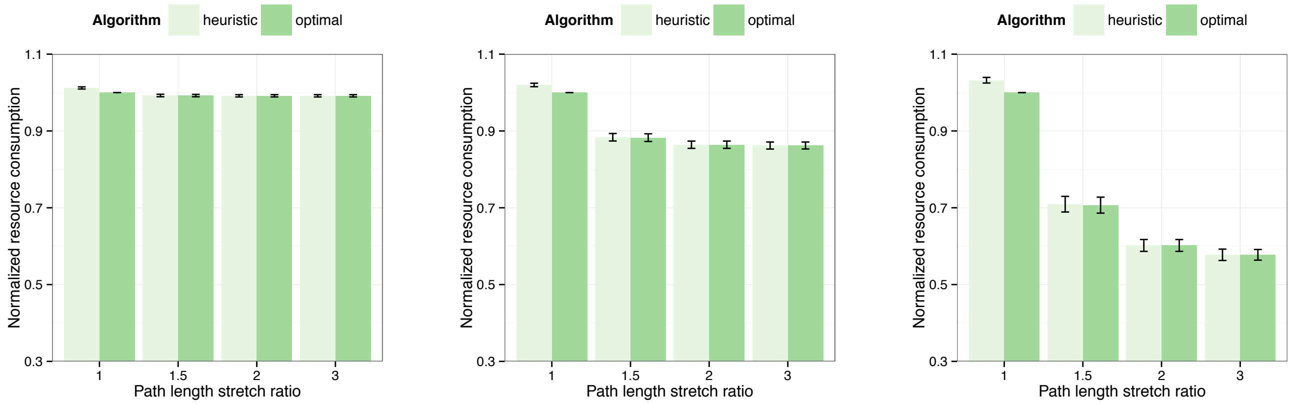
At the end of the main procedure in Algorithm 1, the mirroring switch and the path of original data flow leading to the minimum network resource consumption are selected as the final scheduling result for the flow.

IV. Evaluation

In this section, we report the evaluation of the above described models and heuristic algorithm for computing a resource efficient monitoring strategy. The evaluation is based on extensive numerical simulations. In Subsection IV-A, we describe the evaluation setup. Next, in Subsection IV-B, we introduce the system performance metrics and parameters. Lastly, we present and discuss the results in Subsection IV-C.

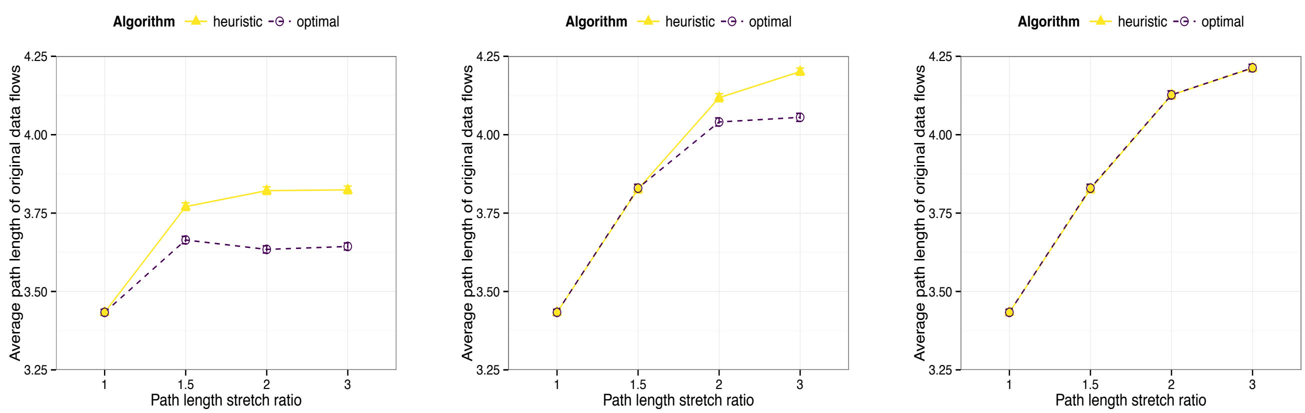
A. Setup

We experiment with a synthetic network generated by FNSS [28], a widely used simulator in the networking community. The network contains 70 switches, and similar to the configuration in [21], its topology follows the Barabasi-Albert model [29]. In addition, we randomly generate 500 original data flows being abstracted as ingress/egress switch pairs and data rates. The data rates followed the standard log-normal distribution to approximate the measured network traffic [30], with a mean value of 10 Mbps and a variance of 0.8 Mbps. We execute the simulations on a machine equipped with a Intel (R) Core (TM)



(a) Link cost ratio = 1 (b) Link cost ratio = 2 (c) Link cost ratio = 5

Fig. 3: Impact of the path stretch ratio and the link cost ratio on the resource consumption



(a) Link cost ratio = 1 (b) Link cost ratio = 2 (c) Link cost ratio = 5

Fig. 4: Impact of the path stretch ratio and the link cost ratio on the path length of original data flows

i5-6500 CPU with four cores and 32GB RAM. We use Gurobi [31] as the optimization solver and repeat each experiment ten times.

B. Evaluation metrics and focused parameters

We use two system performance metrics: (i) the consumed network resources and (ii) the path length of original data flows. The first metric allows to evaluate the benefits gained by our solution, in terms of reduction in the total resource consumption incurred by both original and mirrored data flows. The second metric indicates the penalty brought by our solution, in terms of the change of the path length experienced by original data flows.

In addition, we investigate the impact of the following three parameters on the system performance metrics:

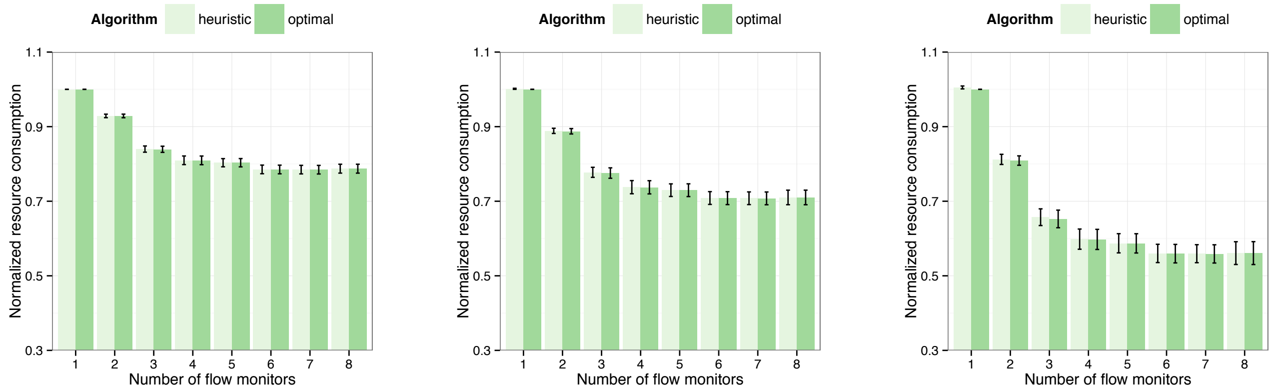
(i) *Path length stretch ratio:* It is defined as the ratio of the maximum allowed path length to the shortest path length. We expect that the larger this value the lower the resource consumption for transmitting the mirrored and original data flows.

(ii) *Number of flow monitors:* It indicates how many flow monitors the monitoring system can utilize. We expect that more flow monitors can reduce the resources consumed in the transmission of data flows.

(iii) *Link cost ratio:* It is defined as the ratio of the cost of transmission in the monitoring plane to the cost of transmission in the data plane. We expect that it is more necessary to stretch the original data flow paths to reduce the resource consumption when increasing this value.

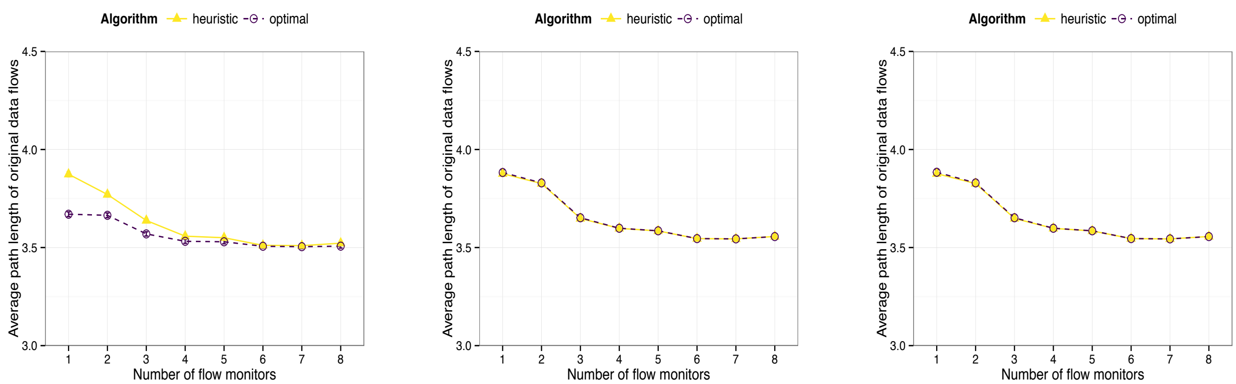
C. Results

1) Impact of the path stretch value and link cost ratio: We fix the number of flow monitors to 2, and vary the path stretch value by selecting it from the discrete value set $\{1, 1.5, 2, 3\}$. The link cost ratio between the monitoring plane and the data plane is chosen from the discrete value set $\{1, 2, 5\}$.



(a) Link cost ratio = 1 (b) Link cost ratio = 2 (c) Link cost ratio = 5

Fig. 5: Impact of the number of flow monitors and the link cost ratio on the resource consumption



(a) Link cost ratio = 1 (b) Link cost ratio = 2 (c) Link cost ratio = 5

Fig. 6: Impact of the number of flow monitors and the link cost ratio on the path length of original data flows

Figure 3 depicts the impact of the path stretch value and the link cost ratio on the resource consumption. In each subfigure, the network resource consumption is normalized by the obtained optimal result when the path stretch value is 1. As shown in Figure 3b and Figure 3c, when the link cost of the monitoring plane is greater than that of the data plane, increasing the path stretch value helps to reduce the overall network resource consumption. More specifically, in the scenario where the link cost ratio is larger, our model and algorithm tend to further push the original data flows closer to the flow monitors to save more resources. For example, allowing the path length of the original data flows to be 0.5 longer than the shortest path can reduce 30% of the network resource consumption when the link cost ratio is 5, compared with 10% resource reduction when the link cost ratio is 2. In addition, we can also see that our heuristic algorithm achieves near-optimal performance, except incurring more resource consumption when the path stretch ratio is 1.

Figure 4 depicts the impact of the path stretch ratio and the link cost ratio on the computed path length of original data flows. As shown in Figure 4b and Figure 4c, increasing the path stretch ratio allows to divert the original data flows away from their shortest path. We can also see that our model and algorithm pushes the original data flows more towards the flow monitors when the link cost ratio increases. For example, increasing the path length stretch ratio from 2 to 3 cannot further reduce resource consumption through path stretching when the link cost ratio is 1 or 2. Our heuristic algorithm generates slightly longer paths for original data flows. This is due to the order of searching for a mirroring switch from the one being close to the flow monitors.

2) Impact of the number of flow monitors and the link cost ratio: We vary the number of flow monitors from 1 to 8, and fix the path stretch ratio to 1.5. We also select the link cost ratio from the discrete value set $\{1, 2, 5\}$.

Figure 5 depicts the impact of the number of flow monitors and the link cost ratio on the resource consumption. In each subfigure, we normalize the network resource consumption with the obtained optimal result when there is only one flow monitor. We can see that, regardless of the link cost ratio,

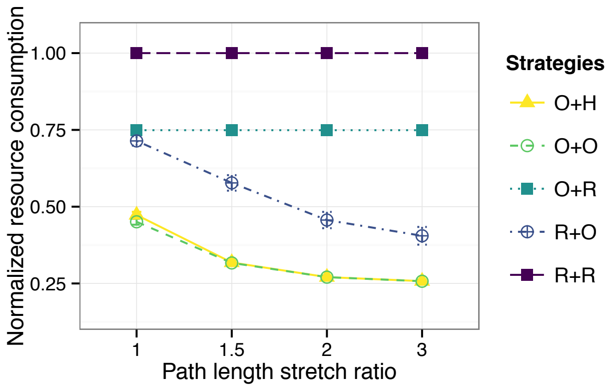


Fig. 7: Performance of difference strategies

increasing the number of flow monitors can always reduce the overall network resource consumption. Particularly, when the cost of transmission in the monitoring plane is higher, stretching the paths of original data flows helps to reduce a large portion of the overall network resource consumption. We can see that increasing the flow monitor number from 1 to 2 or 3 brings the greatest drop in the resource consumption. After that, adding more flow monitors results in only marginal improvement. Again, we can also see that our heuristic algorithm achieves near optimal results. It only performs slightly worse than the optimal results when only a few flow monitors exist.

Figure 6 shows the impact of the number of flow monitors and the link cost ratio on the computed path length of original data flows. In general, we can see that using more flow monitors reduces the paths of original data flows. The reason is that it provides a higher chance that a flow passes through the vicinity of the flow monitors. Compared with the optimal results, our heuristic algorithm computes slightly longer paths of original data flows when the link cost ratio is 1, but achieves almost the same results in the two other scenarios.

3) Baseline strategies comparison: We compare the following strategies:

- Optimal FMPSB + Optimal FEMSS (O+O)
- Optimal FMPSB + Heuristic FEMSS (O+H)
- Random FMPSB + Optimal FEMSS (R+O)
- Optimal FMPSB + Random mirroring switch (O+R)
- Random FMPSB + Random mirroring switch (R+R)

In the random flow monitor placement, we randomly choose the locations of flow monitors and bind the switches with them. In the random mirroring switch selection, original data flows follow the shortest paths, but we randomly select the mirroring switches. We configure the link cost ratio to 5 and assumed two flow monitors in the network. We chose the path stretch ratio from the set $\{1, 1.5, 2, 3\}$. Due to the space constraint, we only show the results of the reduction in the resource consumption achieved with each strategy. The results are normalized by the incurred resource consumption when the system uses the fully random strategy (R+R).

Figure 7 depicts the impact of different strategies on the overall resource consumption. The results of the strategies R+R and O+R remain the same for the different path stretch ratios, since the strategy of randomly selecting the mirroring switches does not need this parameter. The strategy R+R leads to the highest network resource consumption, while our consecutive strategies O+O and O+H, achieve the lowest values, even when only the shortest paths are allowed. Only optimizing the placement of the flow monitors or the flow embedding and mirroring switch selection lead to in-between performance. The results also show that increasing the path length stretch ratio can significantly reduce the overall network resource consumption, even when the flow monitor placement is random.

V. Conclusion and Future work

Mirroring data flows in the network and transmitting the mirrored flows to flow monitors enable in-depth flow packets analysis. However, transmissions of original and mirrored data flows consume network resources. Our main goal in this paper is to reduce the overall costs of transmitting both flow types in distinct communication planes. Towards that end, we present REMO, a network monitoring solution that optimizes the global network resource consumption, in the scenario where the original and mirrored data flows are transmitted in separate planes.

More precisely, in REMO, we use an ILP model to optimize the flow monitor placement and switch binding so that they reside at central locations in the network. We also use a MILP model to optimize the paths to embed original data flows and the selection of their mirroring switches. In order to overcome the computational complexity of the proposed MILP model, we also design a near-optimal heuristic algorithm for the flow embedding and mirroring switch selection sub-problem.

Through an extensive simulation study, we show that REMO is able to dramatically reduce the global resource consumption, compared with several baseline strategies. Stretching the original data flow paths is particularly useful when it costs more resources to transmit data in the monitoring plane. We also show that increasing the number of flow monitors also helps to reduce the global resource consumption and the path length penalty on original data flows. However, this may bring other issues like increasing the resource consumption on the servers hosting more flow monitors. Investigating this problem is among our agenda for future work.

In the future, we plan to integrate the processing and memory limitations of the flow monitors and switches into the model. In addition, exporting flow records, like in IPFIX [32], further reduces the resource consumption but loses the detailed information of the payload data. Depending on the requirement on the granularity of obtained flow information, designing an adaptive network monitoring system is worthy of investigation.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable feedbacks. This work has been supported by the German Research Foundation (DFG) as part of the project A12 and A08 within the Collaborative Research Center (CRC) 912 – HAEC, and the German Society for International Cooperation (GIZ) – project no. 15.2011.3-003.32.

References

[1] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, "I know what your packet did last hop: Using packet histories to troubleshoot networks." in *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, vol. 14, 2014, pp. 71–85.

[2] M. V. Mahoney and P. K. Chan, "Phad: Packet header anomaly detection for identifying hostile network traffic," Tech. Rep., 2001.

[3] M. Kodialam and T. Lakshman, "Detecting network intrusions via sampling: a game theoretic approach," in *Proceedings of the 22th IEEE Conference on Computer Communications (INFOCOM)*, 2003.

[4] S. Yoon, T. Ha, S. Kim, and H. Lim, "Scalable traffic sampling using centrality measure on software-defined networks," *IEEE Communication Magazine*, 2017.

[5] S. Miskovic, G. M. Lee, Y. Liao, and M. Baldi, "Appprint: automatic fingerprinting of mobile applications in network traffic," in *International Conference on Passive and Active Network Measurement*. Springer, 2015, pp. 57–69.

[6] S. Shirali-Shahreza and Y. Ganjali, "Efficient implementation of security applications in openflow controller with flexam," in *Proceedings of the IEEE 21st Annual Symposium on High-Performance Interconnects (HOTI)*, 2013, pp. 49–54.

[7] P. J. Moyer, "Sdn-based mirroring of traffic flows for in-band network analytics," Oct. 20 2015, uS Patent App. 14/918,441.

[8] D. Awduche and Y. Rekhter, "Multiprotocol lambda switching: combining mpls traffic engineering control with optical crossconnects," *IEEE Communications Magazine*, vol. 39, no. 3, pp. 111–116, 2001.

[9] X. Chen, Y.-A. Kim, B. Wang, W. Wei, Z. J. Shi, and Y. Song, "Fault-tolerant monitor placement for out-of-band wireless sensor network monitoring," *Ad Hoc Networks*, vol. 10, no. 1, pp. 62–74, 2012.

[10] P. B. Everdell, C. R. Noel, B. Branscomb, and N. A. Langrind, "Network device including dedicated resources control plane," Aug. 28 2007, US Patent 7,263,597.

[11] D. Halperin, S. Kandula, J. Padhye, P. Bahl, and D. Wetherall, "Augmenting data center networks with multi-gigabit wireless links," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, 2011, pp. 38–49.

[12] K. Han, Z. Hu, J. Luo, and L. Xiang, "Rush: Routing and scheduling for hybrid data center networks," in *Proceedings of the 34th IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[13] R. S. Tucker, R. Parthiban, J. Baliga, K. Hinton, R. W. Ayre, and W. V. Sorin, "Evolution of wdm optical ip networks: A cost and energy perspective," *Journal of Lightwave Technology*, vol. 27, no. 3, pp. 243–252, 2009.

[14] E. Franz, S. Pfennig, B. Matthiesen, C. Scheunert, and E. A. Jorswieck, "Energy models for communication of future computing platforms," in *Proceedings of the 14th IEEE International Conference on Ubiquitous Wireless Broadband (ICUWB)*, 2015.

[15] M. A. Habibi, B. Han, and H. D. Schotten, "Network slicing in 5g mobile communication architecture, profit modeling, and challenges," *arXiv preprint arXiv:1707.00852*, 2017.

[16] T. Shimojo, M. R. Sama, A. Khan, and S. Iwashina, "Cost-efficient method for managing network slices in a multi-service 5g core network," in *Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 1121–1126.

[17] R. Hark, D. Stingl, N. Richerzhagen, K. Nahrstedt, and R. Steinmetz, "Disttm: Collaborative traffic matrix estimation in distributed sdn control planes," in *Proceedings of the 2016 IFIP Networking Conference and Workshops (IFIP Networking)*, 2016, pp. 82–90.

[18] H. Tahaei, R. Salleh, S. Khan, R. Izard, K.-K. R. Choo, and N. B. Anuar, "A multi-objective software defined network traffic measurement," *Measurement*, vol. 95, pp. 317 – 327, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0263224116305759

[19] Z. Su, T. Wang, Y. Xia, and M. Hamdi, "Cemon: A cost-effective flow monitoring system in software defined networks," *Computer Networks*, vol. 92, pp. 101–115, 2015.

[20] H. Chen, L. Li, J. Ren, Y. Wang, Y. Zhao, X. Wang, S. Wang, and S. Xu, "A scheme to optimize flow routing and polling switch selection of software defined networks," *PloS one*, vol. 10, no. 12, p. e0145437, 2015.

[21] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of vdpi functions in nfv infrastructures," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–9.

[22] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varys," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, 2014, pp. 443–454.

[23] O. Polychroniou, R. Sen, and K. A. Ross, "Track join: distributed joins with minimal network traffic," in *Proceedings of the 40th ACM International Conference on Management of Data (SIGMOD)*, 2014.

[24] H. Salah, J. Wulfheide, and T. Strufe, "Coordination supports security: A new defence mechanism against interest flooding in ndn," in *Proceedings of the IEEE 40th Conference on Local Computer Networks (LCN)*, 2015, pp. 73–81.

[25] M. He, A. Basta, A. Blenk, and W. Kellerer, "Modeling flow setup time for controller placement in sdn: Evaluation for dynamic flows," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7.

[26] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[27] X.-N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "Officer: A general optimization framework for openflow rule allocation and endpoint policy enforcement," in *Proceedings of the 34th IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 478–486.

[28] L. Saino, C. Cocora, and G. Pavlou, "A toolchain for simplifying network simulation setup," in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, 2013, pp. 82–91.

[29] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[30] I. Antoniou, V. V. Ivanov, V. V. Ivanov, and P. Zrelov, "On the log-normal distribution of network traffic," *Physica D: Nonlinear Phenomena*, vol. 167, no. 1, pp. 72–85, 2002.

[31] "Gurobi 7.0.2," www.gurobi.com [Accessed 08-September-2017].

[32] B. Claise, "Specification of the ip flow information export (ipfix) protocol for the exchange of ip traffic flow information," 2008.