Optimal Flow Collector Placement In Experimental Networks

Ganesh Chennimalai Sankaran
USC-ISI
Los Angeles, USA
gsankara@isi.edu

Mukund Raghothaman
USC
Los Angeles, USA
raghotha@usc.edu

M. Patrick Collins
USC-ISI
Los Angeles, USA
mcollins@isi.edu

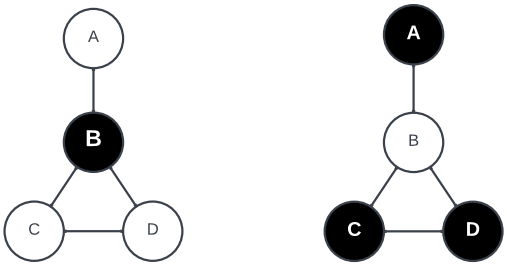
***Abstract*—For operational security personnel, network based data collection requires balancing the need for complete collection against the risks of drowning in redundant data. We have developed a formulation of this problem of redundant collection which converts the problem into a maxflow problem, enabling us to determine optimal sensor placement assuming complete collection. We discuss the formulation and how it can accommodate multiple security requirements and demonstrate its implementation on multiple candidate networks.**

1. Introduction

This paper describes a formulation and optimization technique to identify the best placement of network monitors for *forensic NetFlow*. We define forensic Netflow as NetFlow data collected for the purposes of security, event reconstruction and incident response. NetFlow, originally developed for billing purposes, has evolved into a standard tool for information security analysis [4], [5].

Forensic NetFlow challenges the traditional concept of NetFlow and derived technologies such as sketch [2], [11], which we call *measurement flow*. Measurement flow collects data on network hardware, competes with other processes on the device, and relies on sampling to manage load balancing and data reduction. Forensic flow, by contrast, is focused on finding low-volume high-impact security events, as important hostile traffic often involves a small number of packets. Forensic flow, consequently, cannot rely on sampling for data reduction and load management, leading to the development of "off-box" netflow generators such as YAF [8] or industrial tools such as NetQuest[1], QRadar[2], and Endace[3], all of which generate unsampled NetFlow for security analysis.

Given that the security community requires complete visibility, as demonstrated by the development of hardware and software to process unsampled NetFlow, the challenge faced is managing *complete* and minimizing *redundant* data collection. We can briefly define completeness as the requirement that for any two nodes on a network, if they communicate with each other, then that flow must be recorded. Redundancy refers to the likelihood that, for any two nodes on the network, *multiple* copies of the same flow will be recorded due to satisfying completeness constraints. Figure 1 shows how completeness and redundancy interrelate; in this figure, a black circle indicates that the network node in question is collecting data. Figure 1(i) shows an example of the network with incomplete instrumentation: in this figure, the central node (B) is monitored, but the endpoints aren't, meaning that C – D traffic is not visible to the monitor. Figure 1(ii) shows an example of the same network with redundant monitoring: in this figure, each flow passes through at least one monitoring point, but in the majority of cases, each flow is recorded *twice*. Redundant monitoring increases the amount of data that an analyst has to process, reducing query times, or requires expensive postprocessing to remove the redundant records.



(i) Minimal Instrumentation (ii) Redundant Instrumentation

Figure 1. The C–D link complicates instrumentation and results in redundant collection.

As Figure 1 shows, complete instrumentation guarantees redundancy. If, for example, we remove the monitor at point A in Figure 1(ii), we need a monitor in place at point B to record A – B traffic, at the cost of redundantly recording C – B traffic at points C and B. Removing point B's monitoring results in the same problem. This is a simple network, but the need to monitor C – D traffic specifically introduces a significant overhead. We therefore consider two problems:

- What is the *minimum* redundancy needed in order to ensure complete collection?
- How to process data at the point of instrumentation to reduce redundancy?

All this work assumes that complete instrumentation is a hard requirement.

The primary technical contribution of this paper is a problem formulation which treats monitor placement as a maxflow problem using multiple redundant paths.

1. https://www.netquestcorp.com
2. https://www.ibm.com/qradar
3. https://www.endace.com

In this formulation, we assume that we have complete knowledge of the network and can identify all potential paths between the nodes; this work was originally developed for instrumenting complex networks on Merge[4] testbeds where we have such knowledge. Within those constraints, we can then optimize for the rate assigned to each path within that node, and within the constraint of complete coverage, reduce the replicated paths and consequently redundant data. This solution, once created, can be communicated to a monitor network as a set of path specifications describing what source/destination pairs an individual monitor should collect. This problem is effectively a specialized case of VNF (Virtual Network Function) placement, and by formulating it as a resource allocation problem, we open up other options for optimal instrumentation. In particular, security instrumentation is often dynamic and adversarial, it is often the case that a high-value internal asset requires more extensive and computationally expensive resources which can be split across multiple nodes in order to reduce computational load.

The remainder of this paper is structured as follows: §2 discusses the issue of forensic flow instrumentation and other solutions for monitor placement, §3 discusses the problem formulation and the proposed solution, §4 discusses the results of evaluating our system on test networks on a Merge testbed, and §5 concludes the work.

2. Related Work

NetFlow reporting standards, particularly IPFIX [1] recognize the need for multipoint monitoring by specifying observation points and the idea of observation domains [6]. However, as discussed earlier, optimal flow monitoring has often been secondary due to performance issues with on-switch monitoring [10], [15], leading to a preference for sampling and sketch [2], [11], [12]. This preference for sampling approaches, along with the difficulties involved in network mapping, leads to a broad class of what we term *topology-agnostic* collection techniques, these include systems such as coordinated sampling [13] and OmniMon [7]. These systems use a flow's addressing information to distribute processing across one or more monitors without needing to know how the traffic is routed.

The problem of optimal flow monitor placement is related to the question of ideal NFV placement within SDN, and there exists an extensive literature on SDN-based solutions for monitoring. Lin*et al.* [9] develop a vertex cover based mechanism which exploits architectural features of datacenters, while Thakoor *et al.* [16] develop a FastMap-based mechanism for general placement. Many of the SDN mechanisms, such as Bohatei [3] and NETSECVISOR [14] rely on SDN to reroute traffic to specific locations, whereas our approach assumes routing and instrumentation are separate, an issue in particular when in practice, security personnel have limited control over routing infrastructure.

4. https://mergetb.org

3. System Description

In this section, we present a formal description of the redundant collection problem. Recall from §1 that the goal of monitoring is to achieve *complete* collection with minimal *redundancy*.

We assume that the network is described as a graph, whose vertices are devices and edges are connections between these devices. We define a *monitor* as a process or system which is resident at the same vertex as a device. Monitors may be implemented either as processes on the device, or through sampling tools such as SPAN ports or vampire taps. These monitors can then *observe* traffic at their corresponding vertices, meaning that they can collect and summarize any traffic which the vertex would process either by virtue of being the endpoint for that traffic or by serving as a middlebox to forward that traffic elsewhere in the network. We also assume that monitors are capable of *filtering* traffic, meaning that based on packet attributes, they can opt to observe or not observe a particular flow. Monitoring is entirely vertex based, there is no edge based monitoring.

3.1. Problem Formalization

Consider a network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of its devices (*e.g.*, middleboxes, end user devices and routers) and where $\mathcal{E}$ is the set of its connections. Within this network, a *monitor* is a system present at a device $v \in \mathcal{V}$ which is capable of collecting any packet observed by v. Each monitor can additionally be configured to record some specified subset of packets passing through the device (*e.g.*, based on source and destination devices). The *monitor set*, $\mathcal{M}$ is a set of devices $\{m_1, \dots m_k\}$ where $k = |\mathcal{V}|$ and describes all possible monitors within $\mathcal{G}$. Each monitor m_l is associated with an observation *capacity* cap_l.

A *path* p_{ij} is a sequence of adjacent vertices in $\mathcal{V}$ between source and destination vertices, $v_i, v_j \in \mathcal{V}$. We write $\mathcal{P}(i,j)$ for the set of *all* paths between v_i and v_j, and use subscript notation to differentiate paths within $\mathcal{P}$. For example, p_{ija} and p_{ijb} describe two different paths between v_i and v_j in $\mathcal{P}(i,j)$ For any path p_{ijk}, we associate a *rate* r_{ijk}, which describes the traffic rate mapped to the corresponding path.

A *monitor profile* is a set of monitors locations $\{V_1, \dots, V_k\} \subseteq \mathcal{V}$ together with associated configuration rules for each V_i. A monitor profile is *complete* if every flow is recorded at *at least one monitor*, V_i. The *redundancy* of a collection is the number of repeated records it records because rules in $V_i \text{and} V_j$ result in recording the same flow. Given a topology $\mathcal{G}$, our goal is to create a monitor profile which ensures complete coverage and with minimal redundancy. There are several optimization problems that are of interest:

Minimizing the number of monitoring vertices. In several cases, the analyst is interested in observing flows between devices v_i and v_j at a specific monitor m_l. We therefore begin with a set of Boolean *preselection variables*, ps_{ijkl} indicating whether the monitor m_l has been chosen to observe communication between v_i and v_j that occurs along path p_{ijk}. The monitor placement algorithm, in turn, derives a set of Boolean *selector variables* s_{ijkl}

indicating whether m_l must also be activated to observe v_i–v_j communications occurring along path p_{ijk}. These variables must be chosen so that:

1) Capacity constraints are respected. For each monitor m_l, we require that:

$$\sum_{i,j,k} r_{ijk} \mathbb{1}(ps_{ijkl} \vee s_{ijkl}) \leq cap_l,$$

where $\mathbb{1}(v)$ is the Boolean indicator function, assuming value 1 if $v =$ true and 0 otherwise.

2) Global coverage is achieved. For each pair of distinct source and destination vertices, v_i, v_j, and for all paths p_{ijk}, we require:

$$\bigvee_l ps_{ijkl} \vee s_{ijkl} = \text{true}.$$

3) Flow collections are deduplicated. For each pair of distinct source and destination vertices, v_i, v_j, and for all paths p_{ijk}, we require:

$$\sum_l (\mathbb{1}(ps_{ijkl}) + \mathbb{1}(s_{ijkl})) = 1.$$

Observe that the strict form of the deduplication constraint above, which requires the sum to be exactly equal to 1 automatically implies the global coverage constraint.

Subject to these constraints, we would like to minimize the number of monitors activated for flow collection:

$$\text{minimize:} \sum_l \mathbb{1}(\bigvee_{i,j,k} ps_{ijkl} \vee s_{ijkl}).$$

Accounting for inclusion constraints. Analysts may sometimes wish to deliberately deactivate certain monitors, m_l, from flow collection. Accordingly, they may switch on or switch off an associated Boolean variable, $incl_l$. In order to account for this setting, we modify the coverage and deduplication constraints as follows:

1) For each pair of distinct v_i, v_j, and for each p_{ijk}, we require:

$$\bigvee_l incl_l \wedge (ps_{ijkl} \vee s_{ijkl}) = \text{true}.$$

2) For each pair of distinct v_i, v_j, and for all paths p_{ijk},

$$\sum_l \mathbb{1}(incl_l)(\mathbb{1}(ps_{ijkl}) + \mathbb{1}(s_{ijkl})) = 1.$$

Alternative optimization objectives. In order to accommodate monitor failures, compromise, or downtime, one might instead wish to maximize residual capacity. In the setting of absolute load balance:

$$\text{minimize:} \max_l \sum_{i,j,k} r_{ijk} \mathbb{1}(ps_{ijkl} \vee s_{ijkl}).$$

In the setting of proportional load balance:

$$\text{minimize:} \max_l \sum_{i,j,k} \frac{r_{ijk}}{cap_l} \mathbb{1}(ps_{ijkl} \vee s_{ijkl}).$$

Formulation as a maxflow problem. The main disadvantage in the above formulations is the high cost of invoking an ILP solver. We therefore investigate the following alternative formulation as a maxflow problem. The central idea is to consider the 4-partite graph $G = (V, E)$, where $V = \{v_0, v_f\} \cup \mathcal{M} \cup \mathcal{P}$, and with edges $v_0 \rightarrow m_l$ with capacity cap_l, edges from $m_l \rightarrow p_{ijk}$ whenever device m_l can observe flow p_{ijk}, and with edges from $p_{ijk} \rightarrow v_f$ with capacity r_{ijk}. Of course, although this version of the problem can be solved in polynomial time, one shortcoming is the possibility of fractional monitoring assignments, which may once again be resolved by using an ILP solver.

In this approach, one may achieve proportional load balance by iteratively adjusting the monitoring capacities by a factor of 2, and employing binary search to find the smallest residual capacities that preserve problem satisfiability. In certain circumstances, even faster techniques may be available, such as in the case when at most two monitoring locations are possible for each flow, in which case, fast algorithms for 2-SAT could potentially be used to determine monitor placement.

4. Evaluation

To determine the efficacy of the placement algorithm, we evaluated the *redundancy ratio* of random flow allocations under multiple topologies. We define the redundancy ratio as a continuous value which represents the expected number of flows that will be collected for a particular monitoring configuration under a particular design and load. We generate the redundancy ratio through a series of Monte Carlo simulations where each individual simulation involves a randomly selected monitoring configuration and randomly generate flow configuration. In our tests, a *run* as the calculation of the redundancy ratio under a specific *topology*, *monitoring configuration* and *flow configuration*. This summarizes each run with two independent variables (topology and monitoring ratio), and one dependent variable (redundancy ratio).

The network configuration is a categorical variable comprised of the class and size of the testbed network. The two network classes are described using the graphs shown in Figure 2 and comprise the *ring* and *ring with sites* topologies. These illustrations are simplifications; each class uses a parameterized generator function enabling us to create topologies of arbitrary size(limited to 25, 50 and 150 nodes for these experiments).

The *ring* class topology is an example of a nontrivial network with multiple access points; it consists of an external ring of access points connected to a central mesh of internal routers. This topology demonstrates the problem of redundant data collection, as the internal mesh points are the logical points of collection on the network, but as additional routers are included, the risk of duplication rises. The second topology, *ring with sites*, adds a collection of endpoint subnetworks, these sites increase the complexity of collection by requiring monitoring within the site and consequently forcing redundant collection.

The second independent variable used in the tests is the *monitoring ratio*. The monitoring ratio is the percentage of nodes within the test network which are *activated* for monitoring. Activation means that the node is collecting traffic data and that the node has been configured to record a specific set of paths by the collection algorithm.

The monitoring ratio can vary between $[0, 1]$ with 0 meaning no nodes are activated and 1 meaning all nodes are activated; analyses in this paper have monitoring ratios between 0.3 and 0.95. For any run, the activated nodes are selected randomly.

The dependent variable for the tests is the redundancy ratio. The redundancy ratio is a continuous value between 0 and with no upper limit and is the ratio of flows collected relative to the minimum number of flows for complete collection. A redundancy ratio of 0 means that no flows were collected, while a redundancy ratio of 1 indicates complete collection. Higher redundancy ratios indicated repeated collection, so a redundancy ratio of, for example 1.25, indicates that out of every four flows generated, five were collected with one being a repeat.

Figure 3 shows the results of the experiments using the ring (Figure 3(i)) and ring with sites (Figure 3(ii)) topologies. Each figure consist of three subplots which show the range of observed redundancy ratios for 25, 50 and 150 node topologies. These results are summarized as boxplots.

As Figure 3 shows, we can achieve complete coverage by instrumenting a large number of nodes (a 1.0 redundancy ratio is usually an outlier case for the 0.33 monitoring ratio, but is a median case with 0.6 or above monitoring ratio). However, the optimized and path-aware collection results in a much *lower* redundancy ratio than we would expect when instrumenting so many nodes. For example, 50% instrumentation in the trivial network shown in Figure 1 will result in a minimal redundancy ratio of 1.5, which is far above any of our observed ratios. We note that the need for *complete* collection drives the choice of monitoring ratios. Many of these monitors are final hops or edge networks and comprise a relatively small volume of total traffic. The increase in redundancy rate in Figure 3 is not a linear progression: once a network is completely monitored, the add on effect from additional monitoring is relatively small. The need for extensive instrumentation also demonstrates the value of the optimized path reduction. The results in Figure 3 show that by intelligently choosing the paths on a per-monitor basis, a researcher can keep the overhead from redundant collection manageable.

5. Conclusions

In this paper, we have examined the relationship between *complete* and *non-redundant* network monitoring. We demonstrated that the forensic need for complete instrumentation requires extensive monitor placement, primarily due to the need to handle relatively rare point connections. By intelligently assigning path-specific filtering to individual monitors, we can keep the redundancy ratio manageable and collect complete data.

The results in this paper were scoped specifically to our experimental testbed work, where we have complete visibility into the network under test. There are significant challenges in future work that are related to partial network knowledge. In future work we will build on the ramifications to this problem, including the presence of confounding middleboxes such as NATs or VPN concentrators, dynamic network reconfiguration such as due to a link outages, and how complete the instrumenter's knowledge of the network is.

An interesting question for further analysis and optimization is the value of distinguishing instrumentation on endpoints and edge networks as opposed to transit networks. The need for the massive amount of instrumentation comes from the need to monitor relatively lightly used endpoint networks. NetFlow's use in information security is often as a *supplement* to ground truth tools, particularly EDR – analysts use NetFlow to identify blind spots in EDR instrumentation and to monitor assets for which no EDR is available.

Data Availability

A repository of test topologies is available at https://gitlab.com/mcollins_at_isi/topoaquarium

References

[1] Paul Aitken, Benoît Claise, and Brian Trammell. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011, September 2013.

[2] Peter Clifford and Ioana Cosma. A simple sketching algorithm for entropy estimation over streaming data. In Carlos M. Carvalho and Pradeep Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 196–206, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR.

[3] Seyed K. Fayaz, Yoshiaki Tobioka, Vyas Sekar, and Michael Bailey. Bohatei: flexible and elastic ddos defense. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, page 817–832, USA, 2015. USENIX Association.

[4] Mark Fullmer and Steve Romig. The OSU flow-tools package and CISCO NetFlow logs. In *14th Systems Administration Conference (LISA 2000)*, New Orleans, LA, December 2000. USENIX Association.

[5] Carrie Gates, Michael Collins, Michael Duggan, Andrew Kompanek, and Mark Thomas. More netflow tools for performance and security. In *Proceedings of the 18th USENIX Conference on System Administration*, LISA '04, page 121–132, USA, 2004. USENIX Association.

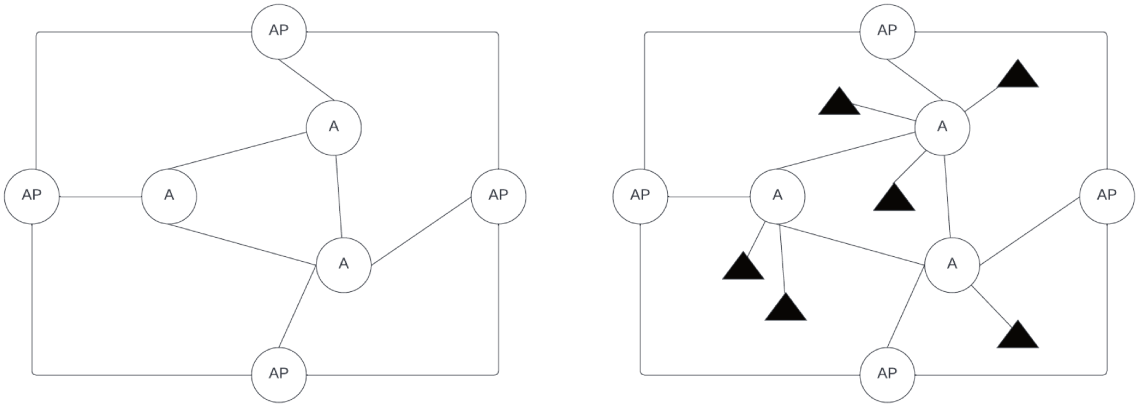
[6] Rick Hofstede, Pavel Čeleda, Brian Trammell, Idilio Drago, Ramin Sadre, Anna Sperotto, and Aiko Pras. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys & Tutorials*, 16(4):2037–2064, 2014.

[7] Qun Huang, Haifeng Sun, Patrick P. C. Lee, Wei Bai, Feng Zhu, and Yungang Bao. Omnimon: Re-architecting network telemetry with resource efficiency and full accuracy. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 404–421, New York, NY, USA, 2020. Association for Computing Machinery.

[8] Christopher M. Inacio and Brian Trammell. Yaf: yet another flowmeter. In *Proceedings of the 24th International Conference on Large Installation System Administration*, LISA'10, page 1–16, USA, 2010. USENIX Association.

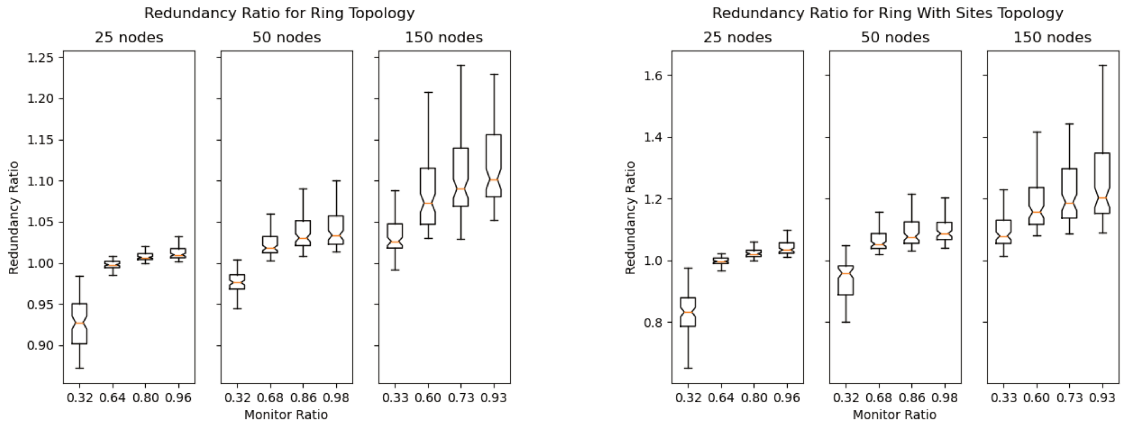
[9] Po-Ching Lin, Chia-Feng Wu, and Po-Hsien Shih. Optimal placement of network security monitoring functions in nfv-enabled data centers. In *2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2)*, pages 9–16, 2017.

[10] Wen-Hong Lin, Wai-Xi Liu, Gui-Feng Chen, Song Wu, Jin-Jiang Fu, Xing Liang, Sen Ling, and Zhi-Tao Chen. Network telemetry by observing and recording on programmable data plane. In *2021 IFIP Networking Conference (IFIP Networking)*, pages 1–6, 2021.



(i) Ring (ii) Ring With Sites

Figure 2. The Ring and Ring With Sites Topologies Are Parametrizable



(i) Loop Topology (ii) Loop With Sites Topology

Figure 3. Keeping Track of Acceptable Paths Reduces Redundancy While Allowing Flows to Be Massively Split Across The Network

[11] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. One sketch to rule them all: Rethinking network flow monitoring with univmon. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 101–114, New York, NY, USA, 2016. Association for Computing Machinery.

[12] Hun Namkung, Zaoxing Liu, Daehyeok Kim, Vyas Sekar, and Peter Steenkiste. SketchLib: Enabling efficient sketch-based monitoring on programmable switches. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 743–759, Renton, WA, April 2022. USENIX Association.

[13] Vyas Sekar, Michael K. Reiter, Walter Willinger, Hui Zhang, and Ramana Rao Kompella. CSAMP: A system for Network-Wide flow monitoring. In *5th USENIX Symposium on Networked Systems Design and Implementation (NSDI 08)*, San Francisco, CA, April 2008. USENIX Association.

[14] Seungwon Shin, Haopei Wang, and Guofei Gu. A first step toward network security virtualization: From concept to prototype. *IEEE Transactions on Information Forensics and Security*, 10(10):2236–2249, 2015.

[15] John Sonchack, Adam J. Aviv, Eric Keller, and Jonathan M. Smith. Turboflow: information rich flow record generation on commodity switches. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, New York, NY, USA, 2018. Association for Computing Machinery.

[16] Omkar Thakoor, Ang Li, Sven Koenig, Srivatsan Ravi, Erik Kline, and T. K. Satish Kumar. The fastmap pipeline for facility location problems. In *PRIMA 2022: Principles and Practice of Multi-Agent Systems: 24th International Conference, Valencia, Spain, November 16–18, 2022, Proceedings*, page 417–434, Berlin, Heidelberg, 2022. Springer-Verlag.