Enhancing Network Attack Detection with Distributed and In-Network Data Collection System

Seyed Mohammad Mehdi Mirnajafizadeh and Ashwin Raam Sethuram, *Wayne State University;* David Mohaisen, *University of Central Florida;* DaeHun Nyang, *Ewha Womans University;* Rhongho Jang, *Wayne State University*

https://www.usenix.org/conference/usenixsecurity24/presentation/mirnajafizadeh

This paper is included in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

Enhancing Network Attack Detection with Distributed and In-Network Data Collection System

Seyed Mohammad Mehdi Mirnajafizadeh
Wayne State University

Ashwin Raam Sethuram
Wayne State University

David Mohaisen
University of Central Florida

DaeHun Nyang
Ewha Womans University

Rhongho Jang
Wayne State University

Abstract

The collection of network data poses a significant challenge for machine/deep learning-driven network defense systems. This paper proposes a new paradigm, namely In-network Serverless Data Collection (ISDC), to eliminate the bottleneck between network infrastructure (where data is generated) and security application servers (where data is consumed). Considering the extremely mismatched scale between traffic volume and in-network resources, we stress the need to prioritize flows based on the application's interests, and a sublinear prediction algorithm is proposed to prioritize specific flows to optimize resource consumption effectively. Additionally, a negotiation-free task migration mechanism with task-data isolation is introduced to allocate tasks dynamically across the network to enhance resource efficiency. Furthermore, ISDC incorporates a serverless data migration and aggregation mechanism to ensure data integrity and serves as a reliable and distributed data source for network defense systems. We present two use cases to demonstrate the feasibility of ISDC, namely covert channel detection and DoS/DDoS attack detection. In both scenarios, ISDC achieves significantly higher flow coverage and feature accuracy compared to existing schemes, leading to improved attack detection accuracy. Remarkably, ISDC's data integrity addresses a model self-poisoning issue caused by duplicated and fragmented flow measurements generated during collaborative measurements.

1 Introduction

Network traffic measurement system is an essential source of data for diverse northbound security applications, including fingerprinting [1, 2], network provisioning [3–5], and anomaly detection [6–17]. Notably, many security applications leverage machine/deep learning (ML/DL) technologies to boost performance [6, 8, 18–22], raising the demands of richer data representations, namely per-flow distribution features [6, 12, 23–26]. However, the measurement and migration of such features come at a high cost, leading to a unique challenge, namely data availability for security (see section 2.1).

Emerging In-Network Computing (INC) technologies enabled flow feature measurements within onsite switches [27, 28] for *real-time* data collection and in-network defenses [6, 12, 15–17, 29–31]. Recent efforts realized measurement of per-flow distribution features using standalone switches [6, 12], focusing on single-point measurement by optimizing resource utilization of data plane functions. Naturally, one can extend single-point measurement solutions [6, 12] to multiple switches to form a distributed flow measurement. Considering imbalanced and dynamic network traffic patterns, a collaborative flow measurement is essential to balance the workload and consolidate resources across the network. Unfortunately, existing collaboration mechanisms [32–34] encounter severe resource inefficiency issues owing to the slow reaction of remote decision-making (i.e., control loop) or limited view of local decision-making (i.e., duplicated tasks).

For learning-based security applications, data migration from distributed switches to a centralized server is the norm, since data aggregation is essential to address data integrity issues (i.e., fragmentation) caused by collaborative flow measurements. However, the data migration triggers unacceptable overheads for northbound links [35–37], which is a critical resource for network operations, but also discourages the collaboration of different network parties due to privacy concerns during raw data sharing. Recent discussions on data migration pay more attention to in-network and real-time convergence of simple counting-based volume measures for the collaborative defense of link flooding attacks (LFA) [13, 14], but not for establishing a data source for ML/DL-based security.

In this paper, we introduce a new paradigm, namely **I**n-network and **S**erverless **D**ata **C**ollection (**ISDC**). As shown in Figure 1, ISDC is a middleware that resides in distributed switches' network operating system (NOS), forming a *data layer*, with the main mission of eliminating the bottleneck between the infrastructure layer and application layer. Interacting with its data plane functions, ISDC is responsible for task allocation and migration to effectively and efficiently utilize in-network resources. Meanwhile, ISDC performs *serverless* data migration and aggregation for data integrity to serve as a

reliable and distributed data source for ML/DL-based security applications.

Our security use cases combine ISDC with the distributed learning concept, delivering various benefits, including eliminating the burden of data migration toward the northbound and opening a possibility for privacy-preserved collaboration among different network organizations. With these insights, we tackle multiple technical challenges to improve ISDC's data collection scalability and quality for security.

Contributions. Our main contributions are as follows:

1. We introduce an in-network data collection framework to enable switch-wise collaboration and engagement of distributed learning techniques for network defense.
2. We propose a sublinear flow size prediction resource-efficient algorithm to prioritize flow measurement.
3. We introduce a negotiation-free task migration mechanism with task-data isolation design to reduce the burden in dynamic task reallocation for efficient utilization of resources across the network.
4. We unveil a data fragmentation issue caused by distributed and collaborative flow measurement, which further triggers a model self-poisoning issue in distributed learning scenarios. Then, we design a data migration and aggregation mechanism for data integrity.
5. We deployed ISDC[1] in a commercial programmable switch and conducted extensive simulations to assess the enhanced data scalability and quality by ISDC.

Two use cases are demonstrated: covert channels and DoS/DDoS attack detection. For covert channel detection, ISDC achieved 94.1% (50.6% higher) flow coverage and 7.8x better feature accuracy compared to state-of-the-art schemes on average, leading to boosted covert attack detection accuracy, F1 0.960 (0.117 higher) and AUC 0.938 (0.176 higher). In DoS/DDoS detection, we unveiled a model poisoning issue caused by duplicated and fragmented measurements. We showed ISDC with data integrity outperforms the state-of-the-art scheme, yielding an AUC of 0.945.

Organization. The rest of the paper is organized as follows. In section 2, we discuss the motivation and challenge for ISDC. Next, we present the ISDC's working flow, functions, and protocols throughout sections 3–5. Then, in section 6, we evaluate system performance and demonstrate security use cases, followed by discussions of system design choices in section 7. Lastly, we discuss related works and conclude our work in sections 8 and 9, respectively.

2 Motivation and Challenges

In this section, we motivate ISDC from a new perspective: data availability (DA) for security. Then, we discuss the technical challenges and design goals of ISDC.

[1]The source code is at https://github.com/NIDS-LAB/ISDC

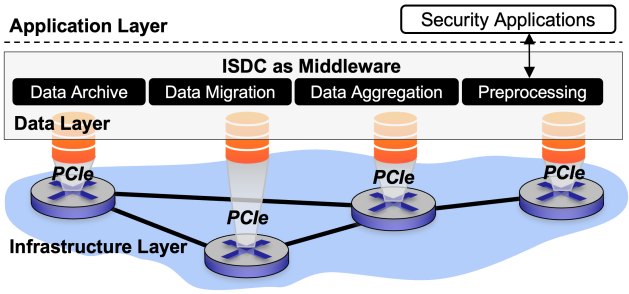
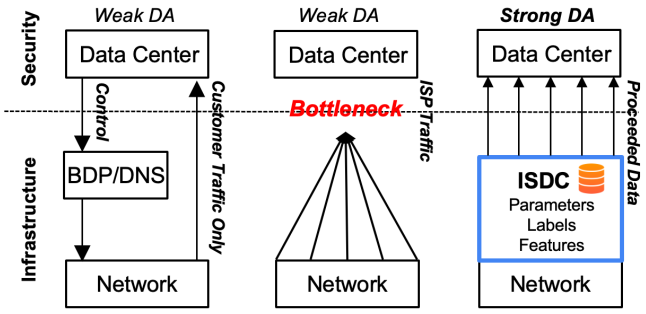


Figure 1: Architecture: ISDC is network storage distributed in network switches. It archives data measured by the switch's data plane functions and performs serverless data migration and aggregation to provide a reliable data source foundation for northbound security applications.

2.1 Data Availability for Security

Cloud Computing-based Defense. The network defense is more challenging than ever. The advance of recent attacks neutralizes signature-based defenses [38], raising demands of advanced ML/DL technologies for defense guidance [22, 35]. Moreover, advanced network devices improve data transmission capacity and empower malicious network parties. Therefore, cloud-based attack mitigation is widely believed to be practical due to the *resource availability* for high-volume traffic processing and heavy ML/DL functions. *However, we stress that a crucial factor of the success is more of its data availability (***DA***) in the cloud.* As illustrated in Figure 2 (a), cloud-based defenses leverage the Border Gateway Protocol (BGP) or Domain Name System (DNS) to direct inbound network traffic to a scrubbing center, which enables continuous data collection for security functions (i.e., DA for security). Nevertheless, the data availability of cloud defense vanishes easily by moving the attack target from endpoints to intermediate infrastructures, e.g., critical network links [13, 39, 40] or Internet Exchange Points (IXPs) [41].

Internet Service Provider (ISP)-level Defense. The weakened data availability of the cloud framework naturally brings our attention to network infrastructure-layer (or ISP-level) solutions. As shown in Figure 2 (b), an ISP with direct access to the in-network switches has the strongest data availability at the infrastructure layer, simply because switches are carriers of network raw data (i.e., packets). However, the major challenge for ISP-level data collection is the data migration from the infrastructure layer to the application layer (i.e., the venue of data consumption) for the following reasons. First, unlike the cloud-based approaches that have a narrow interest in network traffic (i.e., traffic towards customers), the ISP is responsible for the full population of traffic coming through the network with a mission of data transmission, but not complex security due to the vast amount. Second, the northbound bandwidth is reserved for latency-sensitive network operation functions (i.e., policy updates). Continuous data migration towards the northbound adds tremendous burdens to such a



(a) Cloud (b) ISP (c) ISP with ISDC

Figure 2: Motivation: data availability (DA) for security. (a) The cloud-based defense has narrow interests in network traffic, (b) ISP traffic migration towards the application layer is challenging, and (c) ISDC resides in the infrastructure layer and has direct access to network data.

crucial resource and great uncertainties to network control. To these ends, the ISP-level solution with a centralized data collection strategy has weak DA at the application layer.

Vision: In-network Serverless Data Collection (ISDC). The lessons are that 1) the infrastructure layer has the strongest data availability for security by nature and 2) data migration toward the application layer is challenging because of either the inherent nature of weak data availability of the cloud or the large overhead of northbound traffic of ISP, which necessitates a paradigm change of data collection for security applications. With these insights, we propose the concept of in-network serverless data collection (ISDC) with the essential argument that raw data migration towards the northbound is unnecessary. As shown in Figure 2 (c), ISDC resides in the infrastructure layer (i.e., distributed switches) to enjoy strong data availability (DA). Furthermore, instead of migrating raw data towards the northbound, our vision is to migrate partial security functionality from the application layer to the infrastructure layer,e.g., data prepossessing and even local model training. The intuition is that this paradigm change is aligned with the distributed learning paradigm addressing overhead and privacy challenges in terms of data communication [42, 43], as shown in Figure 2 (c). However, multiple technical challenges have to be addressed to achieve ISDC.

2.2 Challenges

Data-Plane Resource Constraint. With adversarial attack patterns, existing in-network security with volume feature [13–15, 17, 44] has shown limitations in identifying application-layer attacks. Therefore, recent works prefer richer data representations, namely per-flow distribution features, for robust attack detection. FlowLens [6] showed the potential of per-flow distribution features in various application-layer attack detections. To accommodate the hardware constraint (i.e., memory and computation) of in-network switches, FlowLens quantized the distribution data leading to fewer per-flow data points, but still showed its feasibility with ML technologies. Nevertheless, the hash table data structure used for per-flow measurement becomes a barrier due to scarce memory space in the switch. Netwarden [12] also uses per-flow distribution data for covert channel detections. It adopted a sketch approach [45] with memory random sharing to scale up per-flow measurement. Unfortunately, the sketch also suffers saturation issues, vanishing its error-bound guarantee. As such, although it is clear that per-flow distribution data is a viable feature, a single-point (switch) measurement encounters severe scalability issues limited by hardware resources.

Resource Wasting during Collaboration. A body of research has explored collaborative and distributed flow measurement leveraging distributed network devices. Coordinated Sampling (CSAMP) [32] and Network-wide Switch Probability Assignment (NSPA) [33] are representative and state-of-the-art works empowered by the global view of the network for optimizing resource utilization. The downside of such approaches is slow adaptation to dynamic shifts in network traffic. Especially considering the massive amount of short-living and mice flows existing in modern networks, massive memory resources will be wasted intermittently during the remote decision-making process (i.e., control loop). Orthogonal to the central management approach, Cooperative Flow Selection (CFS) [34] performs onsite decision-making based on the local view at each switch. Such a strategy eliminates the control loop issue and adapts to dynamic traffic changes timely. However, the limited view causes duplicated measurement tasks launched at multiple switches, triggering memory-wasting issues, as shown in Table 1. The memory inefficiency hurts the data collection scalability, resulting in poor attack detection performance (see section 6.3 for the covert channel detection case study and the associated performance).

Data Fragmentation caused Model Poisoning. Additionally, we report a model poisoning issue when employing a local view-based dynamic task allocation approach [34] as a data source of distributed learning. That is, a single flow can be measured on inclusive or exclusive timelines at different switches on its routing path, which results in the generation of multiple feature fragmentation (i.e., partially measured or overlapped flow features), as shown in Table 1. Especially in a distributed learning system, the presence of such fragmented and inconsistent data can lead to biased local model training and eventually generate a poorly aggregated model (see section 6.4 for DoS/DDoS attack detection performance).

2.3 Design Goals

Table 1 compares the design of existing data collection systems with ISDC. The drawbacks of previous approaches motivated us to design our system with the following goals:

Optimizing Network Resource Usage. Given the finite computational and memory resources in the distributed switches

Measurement Task Coordination				
	CSAMP [32]	NSPA [33]	CFS [34]	**ISDC**
Decision	Remote	Remote	Onsite	Onsite
Task Duplication	✗	✗	✓	✗
Task Priority	✗	✗	✓	✓
Task Migration	✗	✗	✗	✓
Overhead	High	Medium	Very High	Low
Flow Coverage	35.4%	36.9%	58.1%	94.1%
Data Collection				
	CSAMP [32]	NSPA [33]	CFS [34]	**ISDC**
Data Migration	Central	Central	Central	Serverless
Fragmentation	✗	✗	✓	✗
Overhead	High	High	High	Low
Detection AUC	0.718	0.709	0.857	0.938

Table 1: Comparison of designs. The experimental results of our security use case, namely covert channel detection, are shown with two metrics (flow coverage and AUC) to compare the data collection efficiency and quality.

for the flow feature measurement (i.e., tens of Megabyte memory space and tight processing deadline [27, 28]), optimizing the resource utilization is crucial. To do so, we first aim to prioritize flow measurement tasks with security applications' requirements to relax the resource competition in the data plane. Next, we encourage active task migration to address the traffic imbalance and dynamicity in different network nodes to maximize resource usage. Lastly, we target a coordinated task migration to prevent duplicated tasks and resource wastes, as shown in Table 1.

Reliable Data Source for Security. Data integrity is crucial for any data-driven security applications. The traditional approach is to migrate data from the data plane, where the data is generated, to the application servers, where the data is consumed, for data aggregation. However, this decision adds burdens to the northbound links. To overcome this bottleneck, we aim to build a distributed network storage to form a data layer to serve northbound applications with pre-processed data. For data quality, we design a data-layer communication protocol for serverless and in-network data migration to aggregate data fragmentation, as shown in Table 1.

Low-overhead Task/Data Migration. Task and data migration are the primitive functions of ISDC for resource optimization and data integrity. However, the more active these actions are, the more overhead is added to the network links; i.e., trade-off. To tackle this issue, our design goal is to minimize the footprint of task migration and data migration to reduce the burden added to network links, as shown in Table 1.

3 System Design: ISDC

3.1 Framework

As shown in Figure 3, IDSC's data plane functions, namely Flow Identifier (FI) and Feature Meter (FM), are placed in each switch's packet processing pipeline, namely Application-Specific Integrated Circuit (ASIC). IDSC's storage is integrated into the network operating system (NOS) running with offline resources, i.e., general CPU and large DRAM. The communication between data plane functions and storage utilizes a high-bandwidth Peripheral Component Interconnect Express (PCIe) for data archiving. We highlight that ISDC's distributed storage forms a data layer and serves northbound applications.

3.2 End-to-End Design

ISDC system is built on top of distributed network switches with the following essential functions:

❶ **Flow Identification.** As shown in Figure 3, ISDC's first step is identifying targeted flows. Here, we prioritize flows with their sizes, with the key insight that super mice flow, with one or two packets, contains sparse data points of distribution feature is disfavored by ML/DL applications due to difficulty in learning. The flow size is not known beforehand, but the flow feature measurement must start instantly from the first packet. However, predicting flow size in an ad-hoc manner is challenging, especially with a resource constraint. In this work, we propose a Flow Identifier (FI), a simple but efficient flow size prediction algorithm leveraging the burstiness feature to prioritize flows for task allocation (see §4.1).

❷ **Feature Measurement.** Once FI identifies a flow, the flow feature measurement task will be performed at a separate data structure, Feature Meter (FM), a simple hash table. We stress that it is essential for distributed and collaborative flow measurement systems to use a general hash table structure for flow feature measurement since flows must be evictable for task and data migration, for collaborations, and to avoid task duplication (see section 4.2 for details).

❸ **Task Migration.** We stress that narrowing the scope to the targeted flows is insufficient to optimize resource utilization in network switches. Upon collision in FM's hash table, tasks must be handed over (i.e., task migration) to the follow-up switches on a flow's routing path, not to miss feature measurement, as shown in Figure 3. Dynamic task migration allows ISDC to utilize idle resources of switches on flows' routing paths for a collaborative measurement. ISDC's task migration event is triggered by a hybrid eviction policy integrated into FM (see section 5.1 for details).

❹ **Data Archive.** We note that the migrated tasks are often handed over along with the data for an accumulated and continuous measurement [13], which triggers extra in-band (i.e., network link) overhead with frequent task migrations. Our design choice is called *task migration with data isolation*, which archives data at the present switch and then migrates tasks only with negligible overhead (see section 5.2 for details).

❺ **Data Migration.** An issue triggered by task-data isolation is that a flow may be measured at multiple locations *exclusively*, namely flow feature fragmentation. For data aggregation, ISDC migrates all distributed flow features back to their edge switch (i.e., a flow's entry point of the network) [13],

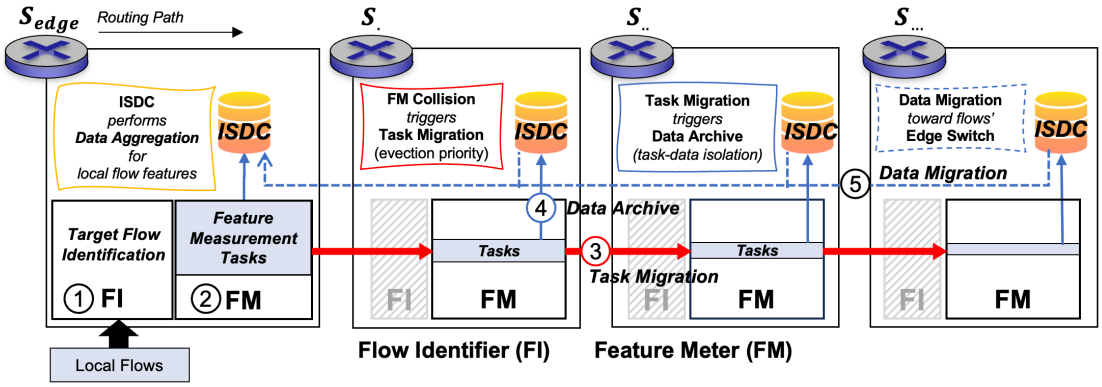


Figure 3: ISDC's workflow: (1) Flow Identifier (FI) is active for local flows only for flow prioritization and target flow identification. (2) Feature Meter (FM) is a simple hash table for per-flow feature measurement of identified flows. (3) FM evicts a task upon collision and migrates the task for collaborative measurement. (4) Data archive is a task-data isolation design to reduce task migration overhead. (5) Data migration is offline towards each flow's edge switch.

combined with the greedy task allocation strategy [34], for minimizing data migration footprints, as shown in Figure 3. Furthermore, we design a novel data migration mechanism, a simple source routing function, to return the flow features to their edge switches. Our data migration is lightweight and works without prior knowledge of network topology (see section 5.3 for further details).

3.3 Distributed Learning/Detection of Attacks

The same processes 1∼5 repeat independently at all edge switches and consistently maintain local flow feature data. With ISDC, network operators can either perform distributed learning of up-to-date traffic or conduct in-network attack detection [6] and mitigation [46], empowered by a federated model with up-to-date knowledge. Furthermore, with the greedy and edge-centered task allocation strategy, attack flows can be detected and mitigated at the network's border (i.e., early intervention), which can help reduce the network link flooding [13]. Our use case combined ISDC with a standard federated learning scheme [43] for covert channel detection and DoS/DDoS attack detection for evaluating the feasibility of our ISDC as a data source for security applications. ISDC data collection and aggregation are performed in the data plane. In our security use cases (see §6.3 and 6.4), ISDC communicates model parameters with a data center but not raw data. However, raw feature data can also be migrated to the northbound for aggregation-free and centralized learning.

4 Function Design

In this section, we describe ISDC's core function designs, including flow size prediction sketch (i.e., Flow Identifier) and feature measurement data structure (i.e., Feature Meter).

4.1 Flow Identifier (FI)

FI addresses a real-time top-K prediction challenge, operating within a very narrow time window and with $O(1)$ memory and computational complexity. Particularly, FI exploits the bursty nature of larger flows occurring within short timeframes, as observed in literature such as [47,48], to forecast deterministic top-K flows for immediate task allocation, which typically demands $O(K)$ counters for post-hoc analysis [44,49]. Burstiness is defined as the continuous arrival of packets with a small inter-arrival interval time (e.g., less than 1 ms).

Data Structure and Algorithm. Figure 4 illustrates FI's data structure and Algorithm 1 outlines its operations. FI maintains a hash table with w slots, each maintaining three variables: a 32-bit Flow ID (key), a 4-bit burst counter (B), and a 4-bit burst counter combining other concurrent flows ($\hat{B}$). As shown in Figure 4, FI distributes flows to w slots permanently for relaxing collisions of concurrent flows, and the follow-up operations fall into five cases. First, when a slot is empty, FI serves in a first-come-first-serve manner, and the first flow is treated as a top-K by recording its Flow ID in the slot (see f_1 in Figure 4). Next, in the event of a flow match, B is increased by one (see f_2). However, if B overflows, B is reset to one, and $\hat{B}$ is reset to zero (see f_3). The packet will be tagged as "targeted" in both events and passed to the feature meter (FM) for feature measurement. Lastly, when a flow is mismatched, $\hat{B}$ is increased by one, and then the packet is tagged as "untargeted", which will not be measured by FM. Instead, a competition will be triggered in the event of $\hat{B}$ update. If $\hat{B} > \lambda \cdot B$ (see f_4) or $\hat{B}$ is saturated (see f_5), the new flow evicts the old flow and becomes a new target.

Analysis. We start with an observation that recent real-world trace CAIDA [50] follows the Zipf distribution, with 87% of flows being mice (i.e., <10 packets). Figure 5 shows the analysis of burst size and flow size using the CAIDA dataset with real timestamps, varying intervals from 100 *us* to 100 *ms*. As shown in Figure 5 (a), with the interval set to 10 *ms*,

Algorithm 1: Flow Identifier (FI)

```
1  Function Flow_Identifier(Packet pkt):
2      key ← H(pkt), idx ← key mod w;
3      item ← {KEY, B, B̂} ← HashTable_FI[idx];
4      if item == null then
5          item ← {key, 1, 0};                          ▷ See f1 in Figure 4
6      end
7      else if item.KEY == key then
8          if item.B<15 then
9              item.B ← item.B + 1;                     ▷ See f2 in Figure 4
10         end
11         else
12             item ← {key, 1, 0};                      ▷ See f3 in Figure 4
13         end
14         Tag_Targeted(pkt, Tag(10_2));
15         Feature_Meter(pkt);
16     end
17     else if item.KEY != key then
18         if item.B̂<15 then
19             item.B̂ ← item.B̂ + 1;
20             if item.B̂/λ == item.B then
21                 item ← {key, 1, 0};                  ▷ See f4 in Figure 4
22             end
23         end
24         else
25             item ← {key, 1, 0};                      ▷ See f5 in Figure 4
26         end
27         Tag_Untargeted(pkt, Tag(00_2));
28     end
```

96% of bursts contain only 1 or 2 packets. Among flows with a small burst size (i.e., ≤ 2 packets), 91% of flows' sizes are smaller than 10 packets and 67% are smaller than 4 packets, as shown in Figure 5 (b). Also can be seen in Figure 5 (a), only 4% of flows have a larger burst size (i.e., > 2 packets). Among these bursty flows, however, only 5% of flows' sizes are smaller than 10 packets, as shown in Figure 5 (b). The results support our assumption that larger flows tend to be more bursty than smaller flows.

Intuition 1: Small λ for Fast Prediction of New Top-K. Based on the above observations, FI's design is grounded on two fundamental assumptions. First, according to our real-world trace analysis, B is filled with super mice in most cases. Second, based on the first assumption, the number of packets of combined counter $\hat{B}$ is approximately the total number of concurrent flows. Therefore, our empirical parameter λ represents the number of flows sharing a counter for averaging combined burstiness (i.e., averaged burstiness $\hat{B}/\lambda$), which approximates the *1-on-1* competition with the current top-K candidate. With this insight, we can expect that FI terminates the burstiness competition within a very small time window by using a small λ=4 (see f_4 in Figure 4).

Intuition 2: Small Counters for Fast Eviction of Old Top-K. Our design takes advantage of small counters for fast takeover and retention of a competing counter B by an active top-K candidate. For example, a new burst flow can quickly build up the $\hat{B}$ to evict inactive flows with a larger burstiness, or a still bursty flow can retain its counter value in B (see f_5 in Figure 4). The reduced counter size also allows FI to enlarge w, reducing the collision of burst flows.

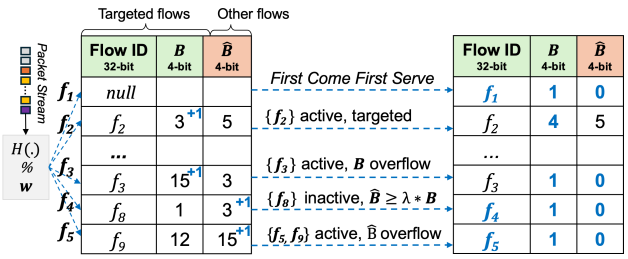
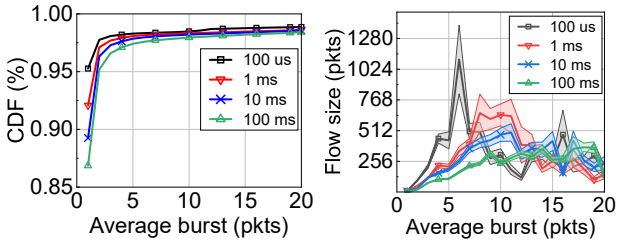


Figure 4: FI operations fall into five cases: a flow takes an empty slot (f_1), the active targeted flow keeps occupying the slot (f_2 and f_3), and concurrent flows evict old flows and become new targeted flows (f_4 and f_5).



(a) CDF of avg. burst (b) Flow size vs. avg. burst

Figure 5: Real-world trace analysis indicates small flows tend to be less bursty than large flows: (a) CDF of averaged flow burst sizes and (b) relationship between flow and burst size.

4.2 Feature Meter (FM)

FM performs per-flow feature measurement tasks for identified flows by FI. We note that the flow feature measurement algorithm is not the main focus of this work. However, we note that it is crucial to employ a scheme based on a general hash table for flow entry eviction and migration, which is essential and straightforward for collaborative flow measurement. In this work, we utilize the feature collection solution of a state-of-the-art scheme, called FlowLens [6], as an add-on module of ISDC system for covert channel detection and DoS/DDoS attack detection. As illustrated in Figure 6, ISDC's FM maintains a simple hash table for per-flow packet size distribution (feature) measurement. According to FlowLens, we quantify the packet size with an interval of 16, resulting in 94 bins. Then, FM records the frequency quantified bins for each flow to represent the packet size distribution (PSD) feature. Each slot (row) in the table records the Flow ID (32-bit), feature distribution, and time-to-live (TTL) value (8-bit).

FM collects features for the FI-predicted top-K flows only, saving scarce memory by ignoring untargeted super mice flows. When a packet of a flow arrives at a slot, FM records the flow feature for the matched flows. If miss-matched, the packet's TTL value is used, with a hybrid eviction policy, to determine which flow to evict for task migration. The evicted flow will be handed over to other switches for collabo-

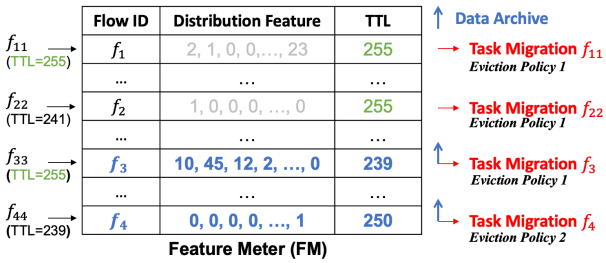


Figure 6: Feature Meter (FM): task migration with a hybrid eviction policy (red) and data archiving event (blue).

rative measurement. In all cases, the eviction policy enforces a higher priority for local over non-local flows, which ensures a flow is measured closer to its edge node to reduce footprint and missing risk when the routing path is short (see section 5.1). Furthermore, a task-data isolation design is proposed for a lightweight task migration (see section 5.2). Eventually, data migration is performed towards edge switches for in-network data aggregation (see section 5.3).

5 Protocol Design

Next, we explain our task migration and data migration mechanism and protocol designs, including the task-data isolation design. Figure 7 shows the customized headers attached to the network packets, namely `Tag` and `Stack`, for task migration and data migration, respectively.

5.1 Task Migration

Event Trigger. To maximize resources utility, we design a task migration (handover) logic based on a simple hash table for collaborative flow feature measurement, as shown in Figure 6. The task migration event is triggered by FM's eviction function with a hybrid policy where FM also has a constant number of slots, and flows are permanently distributed to each slot for resource competition. Our eviction function applies two opposing policies depending on local and non-local flows.

Policy 1 (Local Flow First): Local flows with $TTL = 255$ win the competition and evict non-local flows (i.e., $TTL < 255$). If the existing and newly incoming flows are both local, this policy evicts (handovers) the incoming one to avoid the context switch-caused overhead. This policy aims to retain most flow measurement tasks at the edge switch.

Policy 2 (Lower TTL First for Non-local Flows): When non-local flows compete for the same slot, FM gives a higher priority to lower TTL flows, which is the opposite of Policy 1; i.e., more flows measured before reaching the end of the path.

Protocol. For task migration, ISDC requires only 2-bit tags attached to each packet header, thanks to programmability [51], where the first bit is for FI to tag the targeted flows and the second is for FM to indicate the measured packets and prevent

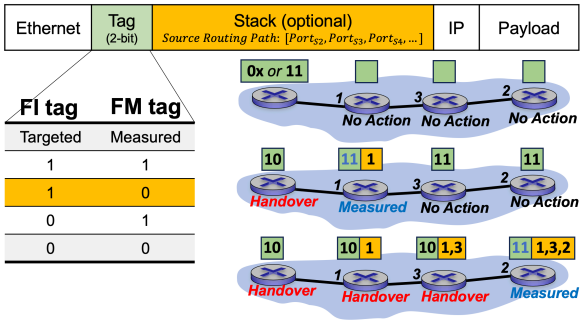


Figure 7: An illustration of task migration with 2-bit tagging. The stack header is only active for packets of unmeasured target flows, with task handover, to record the routing path to the flow's origin (edge) switch for collecting results of migrated tasks (i.e., data migration of fragmentation).

duplicated flow measurement. Figure 7 shows the combination of the 2-bit tags and corresponding taken actions.

(1) `Tag(0x)` → *Untargeted Flows (no action)*: For untargeted flows tagged by FI, all switches on the flow's path bypass the feature measurement and forward to the flow's destination.

(2) `Tag(11)` → *Targeted Flows (measured)*: If a targeted flow has been measured by any FM, no additional action is required for the rest of the switches on the flow's path.

(3) `Tag(10)` → *Targeted Flows (handover)*: If a targeted flow is not measured by its local FM, a task migration event occurs and `Tag(10)` is attached. All switches that observe the tag will allow the flow to compete at their local FM.

To sum up, with the hybrid policy, ISDC aims to maximize the resource utilization of the network while minimizing the footprint of task migrations. With the simple and lightweight tagging mechanism, ISDC's task migration bypasses packets of untargeted flows and measured flows to reduce data plane overhead. Meanwhile, it prevents the duplicated recording of packets from eliminating the overlapping of features for the same flow, which makes data aggregation easier through data migration (see section 6.5 for system evaluation).

5.2 Task-Data Isolation Design

To further reduce task migration overhead, we introduce a task-data isolation design. Upon a task migration event, if the evicted flow has a flow feature measured in FM, the data will not be migrated along with the task but is archived in local and offline DRAM storage through PCIe communication, which significantly saves network bandwidth caused by task migrations (see section 6.5.3 for overhead analysis). Our task migration logic is that once a local flow finds a room at its edge-switch's FM for measurement, it is unlikely to be evicted by other flows and handed over to other switches (i.e., Policy 1). However, as shown in Figure 8 (t_2), if a flow is measured at a non-edge switch (i.e., migrated task), the measurement task can be interrupted and migrated again (i.e., Policy 2), which leads to the fragmentation of flow features distributed

among multiple switches of the flow's routing path, as can be seen in Figure 8 (t_2 and t_4). We note that the distributed and fragmented features for the same flow may trigger model self-pointing issues in the distributed learning scenario (see section 6 for use cases). Therefore, data migration for aggregation is crucial. Lastly, even if a flow is active in the data plane, the flow's data in the control plane can be sent to its edge node immediately due to the task-data isolation design.

5.3 Data Migration

Recall that ISDC aims to be a reliable foundation (data source) with serverless data collection for security applications. To address the feature fragmentation issue, we propose a stack-based source routing to migrate the fragmentation back to the flows' edge switches for data aggregation, namely serverless data collection for data integrity, as demonstrated in Figure 8. The unique challenge posed here is that data migration to the edge often relies on the heavy IP protocol and a strong assumption that every switch has prior knowledge of the entire network topology and routing path [13, 14].

Stack Header for Source Route Tracing. For data migration towards flows' edge switches, ISDC uses stack structure attached to the packet header, which is realized by the built-in function of data plane programming language (i.e., P4 [52]) and supported by commercial programmable switches [53]. Therefore, no extra accommodation is needed to realize the stack function. As shown in Figure 7, the stack header is attached only when a switch observes a packet with unmeasured targeted flows (`Tag(10)`). The functional goal of the stack is to identify the source routing path of a flow at any switch on the flow's path for later data migration towards flows' edge switches (i.e., serverless data aggregation). As shown, whenever a packet reaches a new switch, the switch pushes the packet's incoming port into the stack. As a result, every switch is aware of the physical layer routing path back to the flow's edge (origin) switch by referring to the stacked port numbers. For data migration, as shown in Figure 8, when a switch measures non-local flows, it archives the stack into the local DRAM storage. Therefore, the data migration can be easily achieved by reusing the stack information.

Event Trigger. The data migration is initiated by ISDC actively from the switch's user space of its network operating system. The non-blocking function runs periodically to 1) scan the local DRAM storage, 2) identify inactive non-local flows' data, 3) craft a special packet with the stack as header and data as payload, and 4) eventually inject the packet into the switch's data plane for the source routing.

Understanding the Costs. It is worth mentioning that both source route tracing and data migration are not free but increase network link utilization. The stack size varies according to 1) the number of physical ports available for the switch and 2) the number of task migrations (i.e., hops). For instance,

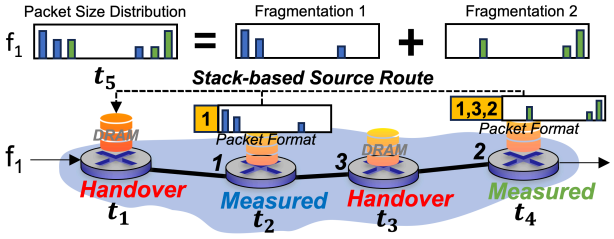


Figure 8: An illustration of data migration using stack-based source routing. A flow f_1 is measured at time t_2 and t_4 by two switches, and each generates a feature fragmentation. The data migration is an active process that sends fragmentation back to the flow's origin (edge) switch for aggregation. The routing path (i.e., incoming switch ports) is collected during task migration with the tag and stack headers.

the widely used switch has 32 ports. Thus, 8-bit is required for each stack slot to distinguish the ports. Moreover, the amount of ports pushed into the stack is determined by the distance of task migration from the flow's edge switch and the non-edge switch that measures the handed-over flow, as shown in Figure 7. However, we reemphasize that the stack is only attached to the unmeasured packets of targeted flows from other switches and eliminated once the packet is measured by any switches. Moreover, combined with the edge-centric task allocation strategy (i.e., Policy 1), ISDC minimizes the network-link overhead of source route tracing for data migration (see section 6.5.3 for overhead analysis).

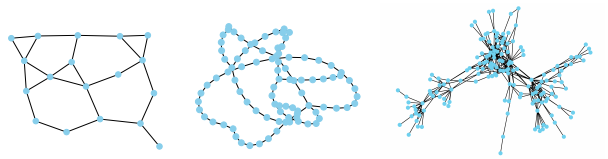
6 Evaluation

This section shows experimental results of ISDC, including security use cases and system performance.

6.1 Experimental Setup

Hardware and Software Implementations. We deployed ISDC prototype on a commercial switch [53] with Intel Tofino-1 Fabric [27] to show the feasibility of ISDC and for function evaluation. The used switch equips 120 MB SRAM and 6.2 MB TCAM memory. Furthermore, we implemented a software version of ISDC with 733 lines of P4 code (P4-16 language [52]) for the data plane and 839 lines of Python code for the control plane. We used `bmv2` model [54] for Mininet simulation [55]. Our simulations were conducted on a server with two 3.2 GHz 20-core CPUs and 512 GB of memory.

Topology. Figure 9 visualizes three network topologies used with Dijkstra routing algorithm [56]. ASN [57] is the smallest topology containing nodes and 25 links and is featured by short but balanced network paths. Vlt Wavenet [57] is a medium-sized topology with 92 nodes and 96 links, featuring significantly imbalanced path lengths. Tiscali [58] is a backbone topology consisting of 161 nodes and 328 links, which is large and complex.



(a) ASN [57] (b) Vlt Wavenet [57] (c) Tiscali (AS 3257) [58]

Figure 9: Real-world topologies used in the experiments.

Dataset and Traffic Generation. We used Facet [59], DeltaShaper [60], and CovertCast [61] for covert channel attacks. We also used CIC-IDS-{2017-2018} [62, 63] and CIC-DDoS-2019 [64] for DoS/DDoS attacks. Moreover, we used an eight-minute CAIDA [50] trace as background traffic from benign clients to add noise during attack data collections. The attack and benign traffic were balanced with the total number of packets. For flawless ML settings, particularly to avoid temporal snooping issues [65], we preserved the order of flows/packets when processing the dataset. Lastly, each network node is attached to a traffic generator for generating traffic using Tcpreplay [66], where all generators send the same number of flows simultaneously without duplication.

Parameters. For fairness, we allocated the same amount of memory space to ISDC and compared schemes [32–34]. Moreover, the total memory budget was equally divided and allocated to each network node. We followed the work guidelines for each switch to configure memory within each network node (switch). Per the guidance, the packet size distribution (PSD) feature is quantized with interval 16 (Quantization Level 4 in FlowLens [6]); thus, 94 bins per flow considering the maximum packet size of 1504 Bytes.

Learning Models. We utilized the Federated Averaging (FedAvg) algorithm [67] to aggregate local models from each node. FedAvg is designed to lessen the impact of training loss on multi-layer neural network [68] in federated learning, where the training data in each node exhibit heterogeneous distributions. Our binary classifier, a simple neural network [69], consists of two hidden layers with 64 and 32 neurons, respectively. These layers utilize the Rectified Linear Unit (ReLU) activation function and incorporate a dropout rate of 20% to prevent overfitting. For decision-making, the network's output layer employs the sigmoid activation function with a threshold of 0.5. Finally, the global model was tested with over 50 runs with an imbalanced ratio of 10/90 for attack/benign samples to reflect real-world settings and reliable results.

6.2 Metrics

Metrics for system and security evaluations are as follows:

Flow Coverage. The coverage quantifies data collection scalability, as $\hat{N}/N$, where $\hat{N}$ is the total number of measured top-K flows and N is the total number of actual top-K flows.

Fragmentation Ratio. Fragmentation ratio is the number of flows measured more than once at different switches (duplicate flows) over the total number of distinct flows. $Frag = \frac{\sum_i \#\text{duplicate flows}}{\sum_i \#\text{distinct flows}}$ for all node i.

Memory Waste. The memory waste is used to quantify the amount of memory wasted for measuring untargeted flows (non-top-K flow in this work). It is defined by U/M, where U is the total number of untargeted flows measured, and M is the total number of slots allocated for the per-flow measurement.

Weighted Mean Relative Error (WMRE). WMRE quantifies data collection quality for per-flow packet size distribution (PSD) features. Accordingly, WMRE in our setting is defined as $\sum_{i=1}^{b} |n_i - \hat{n}_i| / \sum_{i=1}^{b} (\frac{n_i + \hat{n}_i}{2})$, where b is the largest bin, $\hat{n}_i$ is the observed frequency of each bin, and n_i is the ground truth.

F1 Score. We used the F1 score to evaluate attack detection models' performance in our security use cases. It is a harmonic mean of precision and recall to denote the trade-off.

Area Under the ROC Curve (AUC). We further evaluate attack detection models with AUC for a comprehensive measure. The metric summarizes the model performance with various classification thresholds in the ROC curve.

6.3 Use Case 1: Covert Channel Detection

In this experiment, we examine the ISDC's performance with a covert channel detection use case. We first compare ISDC with a strawman approach, a naive extension of FlowLens [6] to a distributed setting, to explain the importance of collaborative and application-focused flow measurement under resource constraints. Then, various state-of-the-art collaboration schemes, such as CSAMP [32], NSPA [33], and CFS [34], were compared to show ISDC's data collection quality. In all use cases, we let each edge switch train a local model independently using locally collected per-flow features, followed by a standard federated learning setting to build a single global model without sending the feature data. Lastly, we conducted an offline evaluation of the global model to measure the impact of the data quality for learning-based covert channel detection. For fairness, we allocated the same amount of memory resources and workloads to all schemes, noting that NSPA, CSAMP, and CFS are real-time decision-making schemes that are directly comparable with ISDC.

Preprocessing. We used 1.1k covert channel attack flows from three datasets [59–61], where 0.88k attack flows were used for data collection simulation and attack model training. Then, the remaining 0.22k attack flows were mixed with 1.98k benign flows to create an imbalanced testing dataset (i.e., 10/90 for attack/benign) to reflect a real-world setting. We note that the model testing is performed offline to measure the impact of data quality in attack detections. Since covert channel attacks are independent, a temporal snooping issue [65] is not a major concern when splitting the flows. Also, we note that these flows have a relatively larger size (e.g., $>$ 1,000 packets). For the background traffic, we matched the packet number with our attack trace from CAIDA benign trace (i.e., 6.5 M packets, 479k flows) with preserved order of packets

and flows. For network topology and traffic generation, we used ANS [57] with 18 nodes, and the mixed attack/benign flows are evenly assigned to all generators without duplication; thus, the traffic imbalance is negligible.

Setting. The memory configuration of all security use cases is determined by the smallest attack flow size in the dataset. In covert channel detection, 76k flows out of 480k total flows are larger than the smallest covert channel flow in size; thus, 76k slots equivalent memory space (i.e., 14.7 Megabyte) was assigned to the entire network. With ANS topology, each node used 836.8 Kilobyte (KB) of memory space. We note that the memory setting is identical for all schemes within the same use case. For ISDC only, 12.2 KB of memory space was taken by FI (1.4% of memory space) for target flow prediction and the remaining was used by FM for per-flow feature data collection. After traffic generation, we used the collected feature data in each node to train a local model and then built a global model from all 18 local models using the standard federated learning framework FedAvg [67]. To analyze the impact of data quality in model training, we conducted experiments with two settings: 1) a single round of training with all nodes involved for a model aggregation and 2) ten rounds of training with 50% nodes randomly selected.

Strawman (Naive) Approach. To enable distributed per-flow feature collection, one naive approach is to deploy FlowLens [6] in a distributed manner, with each switch working independently. We start by showing that this naive deployment, which lacks collaboration and optimization mechanisms, is impractical for overcoming the imbalance of traffic distribution. Table 2 demonstrates the result of the naive approach compared with ISDC. As shown, the strawman approach covers 30.6% flows only among top-76k flows, and 52.5% of memory space was wasted by super mouse flows consisting of one or two packets. Accordingly, the strawman approach achieved a poor F1 score of 0.295 and AUC of 0.246 in the single-round training setting, which is much lower than the ISDC archived F1 score of 0.960 and AUC of 0.938. We noticed that ISDC-based model training converges even with one training round because of the excellent feature engineered in the original work FlowLens (used as FM of ISDC). However, efficient training is unachievable without ISDC's scalable and high-quality data collection. However, we can observe that the naive approach-based model's performance also converges with more training rounds. Still, computation power costs remain inefficient compared to high-quality data-based training.

Compared with Collaboration Schemes. In the following, we used FlowLens [6] feature measurement function as an add-on module to all schemes. Table 2 summarizes the performance of models varying schemes. With a centralized collaboration for task balancing, CSAMP and NSPA improved the flow coverage slightly to 35.4% and 36.9%, respectively, with similar amounts of memory waste. Although unsatisfiable, the central coordination partially addressed the traffic imbalance issue in the network and led CSAMP and NSPA to cover more attack flows in the collected data. As a result, the trained attack detection model achieved higher F1 scores (0.824 and 0.816) and AUCs (0.718 and 0.709), compared to the naive approach (F1 score of 0.295 and AUC of 0.246), in a single-round federated learning setting. As expected, CFS with local but more reactive decision-making achieved a higher coverage up to 58.1% but suffered from low feature accuracy (i.e., WMRE 1.67), mainly because 53% of flows are fragmented, which will be explored more in section 6.4. However, CFS still achieved a higher F1 score of 0.887 and AUC of 0.857, compared to NSPA and CSAMP, in the single-round training setting, which is mainly because of the high coverage of CFS.

Schemes	Cov.	Frag.	Mem. Waste	Avg. WMRE	F1		AUC	
					1 rd.	10 rd.	1 rd.	10 rd.
Strawman	30.6%	0%	52.5%	1.37	0.295	0.927	0.246	0.869
CSAMP	35.4%	0%	51.8%	1.27	0.824	0.923	0.718	0.862
NSPA	36.9%	0%	51.3%	1.25	0.816	0.927	0.709	0.868
CFS	58.1%	53%	62.1%	1.67	0.887	0.942	0.857	0.894
ISDC	**94.1%**	**0%**	**8.02%**	**0.18**	**0.960**	**0.970**	**0.938**	**0.967**

Table 2: Comparing ISDC with a strawman and collaboration schemes in covert channel detection. ISDC, with higher flow coverage (Cov.), superior feature accuracy (averaged WMRE of all flows), and without data fragmentation (Frag.), achieved the best model performance among all schemes.

Lastly, the proposed ISDC achieved the highest flow coverage of 94%, thanks to FI's precise flow size prediction. With a dynamic but duplication-free (fragmentation-free) task migration, ISDC achieved the highest feature quality of WMRE 0.18, as shown in Table 2. Accordingly, ISDC achieved a much higher F1 score of 0.960 and AUC of 0.938, with the same feature and single-round training. An interesting observation is that with a multi-round training setting, the model performance of all schemes improved more or less but was still worse than the ISDC-based model in single-round training, which proves the importance of data quality for training.

6.4 Use Case 2: DoS/DDoS Attack Detection

We further explore ISDC's performance in DoS/DDoS attack detection using CIC-{2017-2019} attack datasets [62–64]. Unlike covert channel trace which mainly consists of large flows, the average flow size in CIC datasets is much smaller (e.g., < 20 packets), requiring more precise feature measurement since missing even one packet may affect overall flow feature quality significantly. With this characteristic, we unveil a model self-poisoning issue caused by a dynamic task allocation [34] and report that data inconsistency for small-sized flows is more critical in model training due to the sparse data points. To verify, we compared ISDC with the state-of-the-art CFS [34], which allocates flow measurement tasks dynamically with local decision-making and achieves the highest flow coverage among existing schemes.

Preprocessing. We randomly selected 10k attack flows from three categories, namely reflection, flooding, and brute force (see appendix B for more details). We used 8k attack flows for simulation and federated learning of attacks. Then, the remaining 2k attack flows were mixed with 18k benign flows to create an imbalanced testing dataset (i.e., 10/90 for attack/benign) to reflect real-world settings. When we split the attack dataset, we preserved time order to avoid a temporal snooping issue [65]. For the background noise, we matched attack traffic with the same number of packets from the benign CAIDA trace (i.e., 2.5 M packets and 232k flows).

Setting. In the mixed dataset, 36k flows are larger than the smallest CIC attack flows in size; thus, 36k slots equivalent memory space was assigned for both ISDC and CFS. With the ASN topology (18 nodes) [57], each node was assigned 395 KB of memory space. For ISDC only, 12.2 KB (3% of memory space) was assigned to FI and the remaining to FM. We used the same federated learning setting as in use case 1.

Compared with Dynamic Decision-Making Scheme. As can be seen in Table 3, CFS achieved a coverage of 49.9%, however, 62% of them are fragmented flow features. To analyze the impact of fragmentation on the model performance, we manually removed the feature fragmentation and retrained the model, denoted as CFS-clean in Table 3. As shown, without fragmentation, the F1 score was improved slightly from 0.617 to 0.620 with the one-round training. Moreover, with ten-round training, the model performance increases significantly in terms of F1 score from 0.613 to 0.756, which infers the model self-poisoning issue is caused by the inconsistency of flow features (i.e., fragmentation) during collaborative flow measurement. An interesting observation is that with the fragmentation elimination, the AUC decreases from 0.828 to 0.777 in the one-round training, which proves our assumption that the fragmentation is more critical for smaller flows; recall that CIC attack flows are mostly less than 20 packets. The results also support ISDC's design of data aggregation, which played an important role in efficient model training. Although the AUC of CFS-clean improves slightly compared to CFS with fragmentation, with ten-round training, the training cost becomes a critical downside.

Lastly, ISDC's fragmentation-free collaboration with flawless aggregation led to a high-quality and scalable feature data collection (see averaged WMREs in Tables 2 and 3), which allows the model to improve the F1 score from 0.730 to 0.809 and AUC from 0.860 to 0.945, with ten-round training setting.

Schemes	Cov.	Frag.	Mem. Waste	Avg. WMRE	F1		AUC	
					1 rd.	10 rd.	1 rd.	10 rd.
CFS	49.9%	62%	67.6%	0.989	0.617	0.613	0.828	0.891
CFS-clean	49.9%	0%	67.6%	0.868	0.620	0.756	0.777	0.892
ISDC	**93.1%**	**0%**	**3.5%**	**0.297**	**0.730**	**0.809**	**0.860**	**0.945**

Table 3: Comparing ISDC with CFS [34], the state-of-the-art dynamic task distribution approach, in DoS/DDoS attack detection. Data duplication and fragmentation in CFS degraded the model's performance, whereas ISDC, with the in-network data aggregation mechanism, boosted the data quality and achieved better model performance.

6.5 System Performance

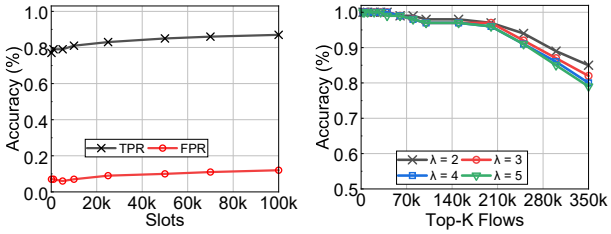
In the following, we use a large topology, namely Tiscali (AS 3257) [58] with 161 switch nodes and 328 links, to evaluate the ISDC's system performance. We first analyze the key function FI in terms of flow prioritization. Next, we verify the data scalability and integrity of ISDC at scale and compare it with other schemes. Then, we systematically evaluate overheads of ISDC, including task/data migration and data archive, to support the feasibility of ISDC. Lastly, we measure the impact of topology by involving two more real-world topologies with different sizes and characteristics (see Figure 9). Worth noting that we show the performance snapshot for all schemes in the progress of the simulation for varied workloads (i.e., from idle to heavy) to compare all schemes fairly.

6.5.1 Flow Prioritization with FI

Recall that FI works with two parameters (w, λ): w is assigned a number of slots and λ is a threshold for prediction. For analysis, we mixed $140k$ attack flows from CIC datasets [62–64] with $\approx$1.8 million benign flows from CAIDA [50]. We use true and false positive rates (TPR and FPR) to show FI's reliability in flow size prediction but not attack detection. TPR correctly predicts top-K flows over the ground truth of top-K, whereas FPR incorrectly predicts top-K flows over the ground truth of non-top-K. Figure 10 shows top-K prediction performance varying w from $0.1k$ to $100k$ with $\lambda = 2$. As shown, the TPR grows quickly as w increases and achieves TPR rates of 81%, 85%, and 87% with $10k$, $50k$, and $100k$ slots, respectively. Remarkably, the TPR of 81% is extraordinary, considering the massive amount of mice flows in the CAIDA trace and the large $K = 350k$ targeted (i.e., the smallest top-K flow has 10 packets only). Moreover, even with a small number of counters (i.e., $w = 10k$), FI shows low FPR, as shown in Figure 10 (a). Furthermore, FI shows stable performance varying λ from 2 to 4, as shown in Figure 10 (b). It is important to note that FI's flow prediction consumes a few packets for burst measure (see appendix A for details), which becomes an inevitable cost for feature measurement in FM. The impact, however, is negligible, as proven by the high feature accuracy in the following section.

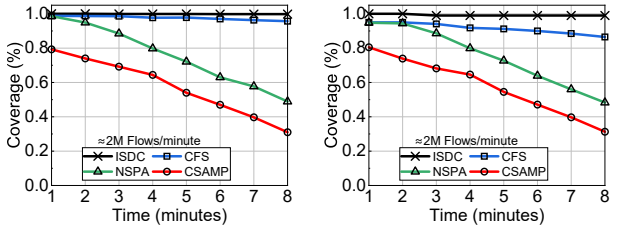
6.5.2 Data Collection Performance

Setting. We used an eight-minute CAIDA benign trace containing around 14.5 million flows. For all compared schemes, we assign 1.1 MB of memory space for each node, which is enough to measure half of the CAIDA trace's flow features. For ISDC, we assign 5 KB of memory space to FI for target



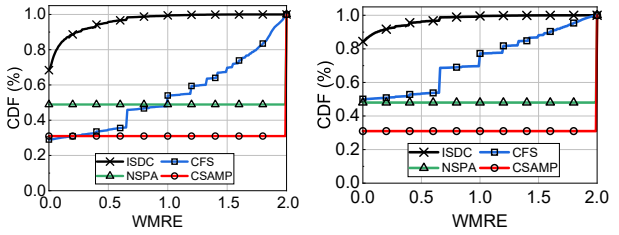
(a) Top-K Prediction varying w (b) Top-K Prediction varying λ

Figure 10: FI Top-K prediction accuracy with CAIDA trace.



(a) Top-100k (b) Top-500k

Figure 11: Comparing flow coverage varying amount of traffic from 1 to 8 minutes of CAIDA traffic. Flow coverage of top-100k and top-500k flows are shown in varying schemes.



(a) Top-100k (b) Top-500k

Figure 12: Comparing feature accuracy of top-K flows varying schemes using Weighted Mean Relative Error (WMRE).

flow prediction and the remaining memory to FM for per-flow feature measurement. For FM, we follow FlowLens [6] standard configuration that measures the most significant ten bins. The system performance snapshots are given at each minute; thus, before the four-minute checking point, data plane resources are enough to measure all flow for all compared schemes. This is to eliminate the impact of FI and show the effectiveness of ISDC's collaboration strategy.

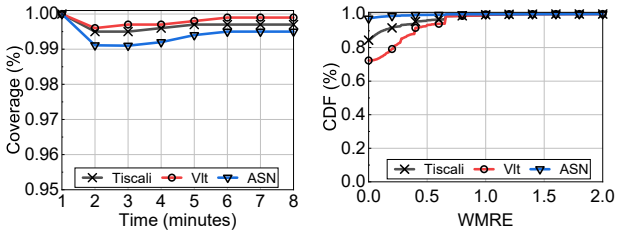
Flow Coverage. Figure 11 shows the coverage of flow feature measurement targeting top-100k and top-500k flows, varying the amount of traffic from 1 to 8 minutes of CAIDA. Notably, ISDC stands out as the top performer, achieving a remarkable full coverage at all times and across all Ks. These results verify the superior performance of FI and ISDC's task migration strategy, which effectively scales up the measurement capacity. On the contrary, the absence of a dynamic collaboration of CSAMP and NSPA on flow measurement resulted in flow omissions when a collision occurs. As can be seen, with dynamic task prioritization, CFS could measure more per-flow features than CSAMP and NSPA. However, the presence of fragmented flows in CFS contributed to approximately 50% of the duplication and inconsistency of flow measurements, further diminishing feature measurement accuracy.

Feature Accuracy. Figure 12 shows the cumulative distribution function (CDF) of WMRE (lower is better) for per-flow feature measurement for top-100k and top-500k flows in the eight minutes of traffic. As illustrated in Figure 11, ISDC easily outperformed other methods by capturing up to 70% more flows in all top-K while achieving WMRE 0 for 60% in top-100k and 80% for top-500k flow feature measurement. Remarkably, 95% of the flows measured by ISDC showed WMRE less than 0.5 among all the top-K, which is clear evidence of the closeness of the measured flow features to ground truth. In contrast, CFS encountering fragmentation issues measured only 35% of the flows with WMRE 0, with a majority exhibiting higher WMRE values. NSPA outperformed CFS in capturing more flows without loss in the top-100k and top-500k flows (see Figure 12). However, both CSAMP and NSPA exhibited high error rates due to flow missing (i.e., WMRE 2), accounting for 50% of flows. To sum up, among all the top-K flows, ISDC achieved an average WMRE of 0.05, while CFS scored 0.54, NSPA 1.04, and CSAMP 1.38.

Topology Impact. We measured the impact of network topologies on ISDC performance. To do so, we conducted experiments using three different real-world network topologies with varied sizes and characteristics (see Figure 9). Figure 13 shows the coverage and the feature quality of top-500k targeted flows for three topologies. ISDC achieved near full coverage consistently across all topologies of more than 99%, which indicates the impact of topologies in ISDC scalability is negligible, as shown in Figure 13 (a). For feature quality, we also observe exceptional accuracy of ISDC across the three topologies, as shown in Figure 13 (b). Even under Vlt topology with extremely imbalanced network path length, ISDC achieved a WMRE of 0 for over 70% flows. In all topologies, over 92% of flows achieved WMREs under 0.5. Therefore, we conclude that ISDC suppresses topology impact well on its stable performance.

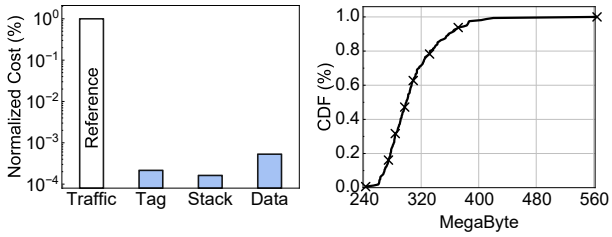
6.5.3 Overhead Analysis

Task and Data Migrations (Network Link). Figure 14 (a) shows the overheads of data and task migrations normalized by the total amount of traffic used for the simulation. The in-band overhead refers to the in-network bandwidth consumption caused by the data and task migrations. The overheads can be categorized into three types: tagging header, stack header, and feature data, where the tagging is for task migration, stack header, and data are for data migration. As shown, ISDC's in-band overhead is negligible compared to the total amount of traffic, considering the tag header is attached to all the packets. Moreover, considering 17% of flow fragmentation generated by ISDC, the overhead triggered by data



(a) Coverage (b) Feature Accuracy

Figure 13: ISDC coverage and feature accuracy comparison for top-500k flow measurement in three real-world topologies.



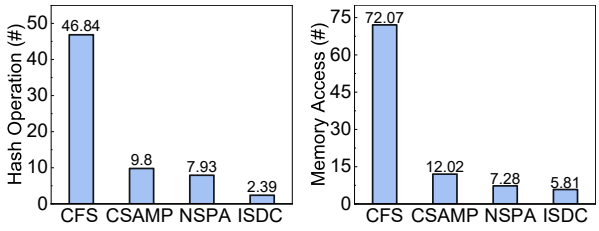
(a) Task-Data Migration (b) Data Archive

Figure 14: Overhead analysis: (a) ISDC's in-network bandwidth consumption for task and data migrations (tag, stack, and data as blue bars), compared with the total amount of network traffic as a reference. (b) CDF of data communication volume through PCIe at all 161 switches for data archiving.

migration (i.e., 0.01% for stack header and 0.05% for data) is also negligible. We can conclude that ISDC adds insignificant overhead to in-network bandwidth resources while eliminating the need for a central collector for data aggregation and saving northbound bandwidth significantly (see appendix A for northbound overhead).

Data Archive (PCIe Bandwidth). Figure 14 (b) illustrates the in-switch communication overhead of ISDC for data archiving from the switch's data plane to the switch's control plane via PCIe. The graph depicts the PCIe utilization of all 161 switches in our large-scale simulation. This overhead arises during the eviction process triggered in the data plane (active data archive) when tasks are transferred to the next switch or when the control plane (passive data archive) releases the data plane memory occupied by those small but bursty flows (short-lived flows). Our analysis revealed that this overhead is minimal compared to the gigabyte capacity of PCIe channels in programmable switches [27, 28]. Remarkably, 90% of switches transfer less than 358 MB during the data archiving process for the eight minutes of CAIDA traffic. This negligible overhead is attributed to the narrowed scope of feature measurement realized by FI's design.

Data Plane Complexity (ASIC). Next, we evaluated the overhead introduced by ISDC on the data plane. Figure 15 depicts ISDC's data plane complexity, comprising per-packet processing tasks such as hash operations and memory access. The computational load in a measurement system is influenced by two key factors: the complexity of data structures in the data plane and the number of hops a packet must traverse to reach the monitoring point. CFS exhibited the highest computational load due to its intricate data plane structure, requiring 11 memory accesses per switch in the worst-case scenario. Although CSAMP and NSPA have less complex data plane designs, they still incur memory access and computation for nearly every packet as it traverses each switch in the path. In contrast, our local eviction policy ensures that most flows are measured at the edge switches. Moreover, our FI design and the tagging mechanism reduce hash computations in subsequent FM switches by bypassing flow feature measurement for all untargeted flows. Consequently, ISDC significantly reduced complexity, resulting in an average of 2.3 hash computations and 5.8 memory accesses.



(a) Hash Operation (per-packet) (b) Memory Access (per-packet)

Figure 15: Data plane complexity: (a) hash operation and (b) memory access required per-packet processing in data plane.

6.5.4 P4 Implementation

We deployed FI and FM into a hardware switch's data plane for feasibility validation. Besides, we used the standard BFRuntime API of Tofino [27] in the switch's control plane to interact with our data plane functions, especially for ISDC's data archiving. We highlight that FI and FM used SRAM memory only and advanced Ternary Content Addressable Memory (TCAM) is not required.

Implementation Issues. We encountered two issues with FI implementation. The first issue is that the hardware switch does not support multiple variables in the same logical structure. Particularly, each slot of FI has three variables, namely the Flow ID (*key*), burst count (B), and combined burst count ($\hat{B}$). To workaround, we adopted the same approach outlined in [44] by approximating $\hat{B}$ with B^{all}. By doing so, B^{all} tracks both B and $\hat{B}$ and counting them simultaneously. Accordingly, the eviction policy will change when the condition $\frac{B^{all}}{\lambda} == B$. The second issue is that the length of variables cannot be varied. Recall that three variables of FI maintain two different bit lengths; thus, we had to match the bit length with the largest variable (*key*). Although this hardware limitation causes some bit waste and is not being used, the total unused memory is negligible, considering the small size of FI.

Hardware Resource Utilization. Table 4 shows the hardware resource utilization of ISDC breakdown with components. Notably, FM that is based on FlowLens [6] exhibits the highest resource utilization, consuming 6.25% of SRAM and

Resources	Flow Identifier	Feature Meter	Protocol	ALL
Stages	3	11	4	12
VLIWs (%)	6.25	3.12	6.25	7.03
ALU (%)	16.67	20.45	0.00	22.92
SRAM (%)	2.50	6.25	0.31	7.19

Table 4: Hardware switch resource utilization of ISDC.

20.45% of Arithmetic Logical Units (ALUs), making it the most resource-intensive element in the system. Conversely, FI primarily relies on computational resources, including Very Long Instruction Words (VLIWs) and ALUs, with minimal SRAM memory and zero TCAM usage. The protocol component also handles header computations for reporting and measurement processes, utilizing 6.25% VLIWs. In summary, ISDC utilizes 7.19% of SRAM, with the majority allocated to FM, along with 2.50% SRAM for FI, in total 22.92% ALUs and 7.03% VLIWs. All these components are successfully fitted into 12 stages of the switch, which infers the line-rate packet processing capacity of ISDC's data plane.

7 Discussions

Impact of Attack Characteristics. We consider two types of attacks: bursty and slow attacks. Bursty attacks exploit the resource constraints of hardware switches by sending numerous burst flows to saturate the data plane. Thanks to ISDC's collaborative and application-focused measurement, saturating all switches across the flow routing path is not trivial. A potential downside is that, with FI design, ISDC mainly considers large and bursty flows. Therefore, adversaries may launch a slow or less bursty attack to evade FI's identification as targeted flows, which we will explore in future research.

Feature and ML Algorithm Selection. The proposed ISDC aims for a scalable and high-quality data collection of richer flow distribution features concerning the resource scarcity of measurement systems but not feature engineering. We opted for FlowLens [6] as our feature extraction module (FM) due to its generalizable hash table-based approach and versatility capable of capturing per-flow features, such as packet size distribution (PSD) and inter-packet delay (IPD). Recently, such distribution features received much attention due to their ability to catch application layer behavior reflected in higher-dimension representation, which benefits attack detections significantly [6, 12, 23, 25, 26]. With the high-quality and rich feature support by ISDC, various advanced ML/DL algorithms [6, 8, 18–22] can be integrated for advanced detection of more sophisticated attacks without a compatibility issue.

8 Related Work

Programmable switches have been used for various applications such as network traffic measurement [29, 44, 70–77] and attack detection [6, 12–17, 25, 30, 46]. Typically, these works concentrate on the switch's inner functionality but often face scalability challenges. Software-defined networking (SDN) is one of the viable approaches for distributed traffic measurement and has been widely discussed [78–83]. However, despite the dynamic nature of SDN solutions, these works suffer from bandwidth overhead and latency issues for measurement tasks [17]. Recent efforts in distributed traffic measurements leveraging programmable switches have been explored for tailored security applications, namely mitigating link flooding attacks [13, 14]. These works focused on consolidating in-network resources for collaborative measurement for fast attack mitigation, focusing on the fast convergence of distributively measured volume data for instant decision-making. Unfortunately, the aim is mismatched with the recent trend of using machine learning technologies with complex features to defeat advanced work. Notably, a large body of research has used ML/DL for network security. These works [8, 23, 26, 84–86] primarily focus on feature engineering and the construction of dedicated models for identifying malicious traffic. Our work is closely related to network-wide task distribution and resource allocation schemes, such as CSAMP [32], NSPA [33], and CFS [34], which aim to achieve full flow measurement coverage by maximizing resource utilization.

9 Conclusion

This paper introduces ISDC, an in-network serverless data collection system for enhancing network defense. ISDC's goal is to serve as a data layer setting in the middle of infrastructure and application layers, aiming to alleviate the bottleneck associated with data flow towards security applications in the northbound. With this objective, we introduced various designs to enhance distributed and collaborative measurement flow features, specifically focusing on consolidating and efficiently utilizing distributed in-network resources. We propose a serverless data migration approach to enable in-network data aggregation, ensuring data integrity for more resilient security applications. Through ISDC, we aim to inspire researchers to integrate advanced machine learning and deep learning technologies into large-scale network attack defense.

Acknowledgements

The authors thank the anonymous shepherd and reviewers for their valuable feedback. This work is supported by the National Science Foundation (NSF) under grant IIS-2211897, by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) No. RS-2023-00222385 and No. 2023R1A2C2006373, and generously by the College of Engineering and Computer Science at UCF.

References

[1] T. Pan, X. Guo, C. Zhang, J. Jiang, H. Wu, and B. Liu, "Tracking millions of flows in high speed networks for application identification," in *Proc. of IEEE INFOCOM*, 2012.

[2] M. Liberatore and B. N. Levine, "Inferring the source of encrypted HTTP connections," in *Proc. of ACM CCS*, 2006.

[3] R. Soulé, S. Basu, P. J. Marandi, F. Pedone, R. D. Kleinberg, E. G. Sirer, and N. Foster, "Merlin: A language for provisioning network resources," in *Proc. of ACM CoNeXT*, 2014.

[4] A. Gupta, J. M. Kleinberg, A. Kumar, R. Rastogi, and B. Yener, "Provisioning a virtual private network: a network design problem for multicommodity flow," in *Proc. of ACM STOC*, 2001.

[5] R. Meier, P. Tsankov, V. Lenders, L. Vanbever, and M. T. Vechev, "Nethide: Secure and practical network topology obfuscation," in *Proc. of USENIX Security*, 2018.

[6] D. Barradas, N. Santos, L. Rodrigues, S. Signorello, F. M. V. Ramos, and A. Madeira, "Flowlens: Enabling efficient flow classification for ml-based network security applications," in *Proc. of ISOC NDSS*, 2021.

[7] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "Pingpong: Packet-level signatures for smart home device events," *arXiv:1907.11797*, 2019.

[8] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. of ISOC NDSS*, 2018.

[9] Y. Xie, V. Sekar, D. A. Maltz, M. K. Reiter, and H. Zhang, "Worm origin identification using random moonwalks," in *Proc. of IEEE S&P*, 2005.

[10] B. Anderson and D. A. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proc. of ACM AISec@CCS*, 2016.

[11] P. Narang, S. Ray, C. Hota, and V. Venkatakrishnan, "Peershark: Detecting peer-to-peer botnets by tracking conversations," in *Proc. of IEEE SPW*, 2014.

[12] J. Xing, Q. Kang, and A. Chen, "Netwarden: Mitigating network covert channels while preserving performance," in *Proc. of USENIX Security*, 2020.

[13] H. Zhou, S. Hong, Y. Liu, X. Luo, W. Li, and G. Gu, "Mew: Enabling large-scale and dynamic link-flooding defenses on programmable switches," in *Proc. of IEEE S&P*, 2023.

[14] J. Xing, W. Wu, and A. Chen, "Ripple: A programmable, decentralized link-flooding defense against adaptive adversaries," in *Proc. of USENIX Security*, 2021.

[15] M. Zhang, G. Li, S. Wang, C. Liu, A. Chen, H. Hu, G. Gu, Q. Li, M. Xu, and J. Wu, "Poseidon: Mitigating volumetric ddos attacks with programmable switches," in *Proc. of ISOC NDSS*, 2020.

[16] Q. Kang, L. Xue, A. Morrison, Y. Tang, A. Chen, and X. Luo, "Programmable in-network security for context-aware byod policies," in *Proc. of USENIX Security*, 2020.

[17] Z. Liu, H. Namkung, G. Nikolaidis, J. Lee, C. Kim, X. Jin, V. Braverman, M. Yu, and V. Sekar, "Jaqen: A high-performance switch-native approach for detecting and mitigating volumetric ddos attacks with programmable switches," in *Proc. of USENIX Security*, 2021.

[18] A. Abusnaina, A. Khormali, D. Nyang, M. Yuksel, and A. Mohaisen, "Examining the robustness of learning-based ddos detection in software defined networks," in *Proc. of IEEE DSC*, 2019.

[19] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *Proc. of IEEE SPW*, 2018.

[20] X. Yuan, C. Li, and X. Li, "Deepdefense: identifying ddos attack via deep learning," in *Proc. of IEEE SMARTCOMP*, 2017.

[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. of ICISSP*, 2018.

[22] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, and L. Cavallaro, "INSOMNIA: towards concept-drift robustness in network intrusion detection," in *Proc. of AISec@CCS*, 2021.

[23] C. Fu, Q. Li, and K. Xu, "Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis," in *Proc. of ISOC NDSS*, 2023.

[24] M. Nasr, A. Houmansadr, and A. Mazumdar, "Compressive traffic analysis: A new paradigm for scalable traffic analysis," in *Proc. of ACM SIGSAC*, 2017.

[25] G. Zhou, Z. Liu, C. Fu, Q. Li, and K. Xu, "An efficient design of intelligent network data plane," in *Proc. of USENIX Security*, 2023.

[26] C. Fu, Q. Li, M. Shen, and K. Xu, "Realtime robust malicious traffic detection via frequency domain analysis," in *Proc. of ACM CCS*, 2021.

[27] Intel, "Tofino-1." [Online]. Available: https://www.intel.com/content/www/us/en/products/details/network-io/intelligent-fabric-processors/tofino.html

[28] ——, "Tofino-2." [Online]. Available: https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-2-series.html

[29] X. Chen, S. L. Feibish, M. Braverman, and J. Rexford, "Beaucoup: Answering many network traffic queries, one memory update at a time," in *Proc. of ACM SIGCOMM*, 2020.

[30] S. Kim, C. Jung, R. Jang, D. Mohaisen, and D. Nyang, "A robust counting sketch for data plane intrusion detection," in *Proc. of ISOC NDSS*, 2023.

[31] A. Agarwal, Z. Liu, and S. Seshan, "Heterosketch: Coordinating network-wide monitoring in heterogeneous and dynamic networks," in *Proc. of USENIX NSDI*, 2022.

[32] V. Sekar, M. K. Reiter, W. Willinger, H. Zhang, R. R. Kompella, and D. G. Andersen, "cSamp: A system for network-wide flow monitoring," in *Proc. of USENIX NSDI*, 2008.

[33] H. Xu, S. Chen, Q. Ma, and L. Huang, "Lightweight flow distribution for collaborative traffic measurement in software defined networks," in *Proc. of IEEE INFOCOM*, 2019.

[34] R. B. Basat, G. Einziger, and B. Tayh, "Cooperative network-wide flow selection," in *Proc. of IEEE ICNP*, 2020.

[35] D. Han, Z. Wang, W. Chen, K. Wang, R. Yu, S. Wang, H. Zhang, Z. Wang, M. Jin, J. Yang, X. Shi, and X. Yin, "Anomaly detection in the open world: Normality shift detection, explanation, and adaptation," in *Proc. of ISOC NDSS*, 2023.

[36] O. Gheibi and D. Weyns, "Lifelong self-adaptation: Self-adaptation meets lifelong machine learning," in *Proc. of ACM SEAMS*, 2022.

[37] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "Cade: Detecting and explaining concept drift samples for security applications." in *Proc. of USENIX Security*, 2021.

[38] C. Zhang, X. Costa-Pérez, and P. Patras, "Adversarial attacks against deep learning-based network intrusion detection systems and defense mechanisms," *IEEE/ACM Trans. Netw.*, vol. 30, no. 3, pp. 1294–1311, 2022.

[39] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *Proc. of IEEE S&P*, 2013.

[40] J. M. Smith and M. Schuchard, "Routing around congestion: Defeating ddos attacks and adverse network conditions via reactive BGP routing," in *Proc. of IEEE S&P*, 2018.

[41] Cloudflare, "The ddos that almost broke the internet." [Online]. Available: https://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet

[42] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, 2020.

[43] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv:1610.0549*, 2016.

[44] T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig, "Elastic sketch: adaptive and fast network-wide measurements," in *Proc. of ACM SIGCOMM*, 2018.

[45] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, 2005.

[46] C. Jung, S. Kim, R. Jang, D. Mohaisen, and D. Nyang, "A scalable and dynamic ACL system for in-network defense," in *Proc. of ACM CCS*, 2022.

[47] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. of ACM IMC*, 2010.

[48] S. Kandula, D. Katabi, S. Sinha, and A. W. Berger, "Dynamic load balancing without packet reordering," *Comput. Commun. Rev.*, vol. 37, no. 2, 2007.

[49] R. B. Basat, X. Chen, G. Einziger, R. Friedman, and Y. Kassner, "Randomized admission policy for efficient top-k, frequency, and volume estimation," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1432–1445, 2019.

[50] CAIDA, "The cooperative association for internet data analysis, equinix chicago data center," 2018, [13:00-14:00, Apr 19 2018., from Sao Paulo to New York]. [Online]. Available: https://www.caida.org

[51] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: programming protocol-independent packet processors," *Comput. Commun. Rev.*, vol. 44, no. 3, 2014.

[52] O. N. Foundation, "P4 language and related specifications." [Online]. Available: https://p4.org/specs

[53] Edge-Core, "Wedge100s-32x data center switch." [Online]. Available: https://www.edge-core.com/productsInfo.php?cls=1&cls2=5&cls3=181&id=382

[54] O. N. Foundation, "P4 behavioral model," https://github.com/p4lang/behavioral-model.

[55] Mininet, "An instant virtual network on your laptop (or other pc)." [Online]. Available: http://mininet.org

[56] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, 1959.

[57] "The internet topology zoo." [Online]. Available: http://www.topology-zoo.org/

[58] N. T. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," in *Proc. of ACM SIGCOMM*, 2002.

[59] S. Li, M. Schliep, and N. Hopper, "Facet: Streaming over videoconferencing for censorship circumvention," in *Proc of ACM WPES*, 2014.

[60] D. Barradas, N. Santos, and L. E. Rodrigues, "Deltashaper: Enabling unobservable censorship-resistant tcp tunneling over videoconferencing streams." *Proc. Priv. Enhancing Technol.*, vol. 2017, no. 4, pp. 5–22, 2017.

[61] R. McPherson, A. Houmansadr, and V. Shmatikov, "Covertcast," *Privacy Enhancing Technologies*, vol. 3, pp. 1–14, 2016.

[62] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Intrusion detection evaluation dataset (cic-ids2017)," 2017. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[63] ——, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. of ICISSP*, 2018.

[64] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *Proc. of ICCST*, 2019.

[65] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *Proc. of USENIX Security*, 2022.

[66] Appneta, "Tcpreplay." [Online]. Available: https://tcpreplay.appneta.com/

[67] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of AISTATS*, 2023.

[68] B. Song, P. Khanduri, X. Zhang, J. Yi, and M. Hong, "FedAvg converges to zero training loss linearly for over-parameterized multi-layer neural networks," in *Proc. of ICML*, 2023.

[69] M. Jiang, C. Hou, A. Zheng, S. Han, H. Huang, Q. Wen, X. Hu, and Y. Zhao, "Adgym: Design choices for deep anomaly detection," *arXiv:2309.15376*, 2023.

[70] A. Gupta, R. Harrison, M. Canini, N. Feamster, J. Rexford, and W. Willinger, "Sonata: Query-driven streaming network telemetry," in *Proc. of ACM SIGCOMM*, 2018.

[71] J. Sonchack, O. Michel, A. J. Aviv, E. Keller, and J. M. Smith, "Scaling hardware accelerated network monitoring to concurrent and dynamic queries with *flow," in *Proc. of USENIX ATC*, 2018.

[72] S. Narayana, A. Sivaraman, V. Nathan, P. Goyal, V. Arun, M. Alizadeh, V. Jeyakumar, and C. Kim, "Language-directed hardware design for network performance monitoring," in *Proc. of ACM SIGCOMM*, 2017.

[73] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, "Heavy-hitter detection entirely in the data plane," in *Proc. of the Symposium on SDN Research*, 2017.

[74] R. Harrison, Q. Cai, A. Gupta, and J. Rexford, "Network-wide heavy hitter detection with commodity switches," in *Proc. of the Symposium on SDN Research*, 2018.

[75] Y. Zhao, K. Yang, Z. Liu, T. Yang, L. Chen, S. Liu, N. Zheng, R. Wang, H. Wu, Y. Wang, and N. Zhang, "Lightguardian: A full-visibility, lightweight, in-band telemetry system using sketchlets," in *Proc. of USENIX NSDI*, 2021.

[76] H. Namkung, Z. Liu, D. Kim, V. Sekar, and P. Steenkiste, "SketchLib: Enabling efficient sketch-based monitoring on programmable switches," in *Proc. of USENIX NSDI*, 2022.

[77] D. Dao, R. Jang, C. Jung, D. Mohaisen, and D. Nyang, "Minimizing noise in hyperloglog-based spread estimation of multiple flows," in *Proc. of IEEE DSN*, 2022.

[78] Y. Yu, C. Qian, and X. Li, "Distributed and collaborative traffic monitoring in software defined networks," in *Proc. of ACM HotSDN*, 2014.

[79] X. Wang, X. Li, S. Pack, Z. Han, and V. C. M. Leung, "STCS: spatial-temporal collaborative sampling in flow-aware software defined networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, 2020.

[80] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "Opennetmon: Network monitoring in openflow software-defined networks," in *Proc. of IEEE NOMS*, 2014.

[81] X. Yu, H. Xu, D. Yao, H. Wang, and L. Huang, "Countmax: A lightweight and cooperative sketch measurement for software-defined networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, 2018.

[82] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, "DREAM: dynamic resource allocation for software-defined measurement," in *Proc. of ACM SIGCOMM*, 2014.

[83] Y. Mi, D. Mohaisen, and A. Wang, "Autodefense: Reinforcement learning based autoreactive defense against network attacks," in *Proc. of IEEE CNS*, 2022.

[84] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, and R. Ranjan, "A hybrid deep learning-based model for anomaly detection in cloud datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 924–935, 2019.

[85] L. Invernizzi, S. Miskovic, R. Torres, C. Kruegel, S. Saha, G. Vigna, S. Lee, and M. Mellia, "Nazca: Detecting malware distribution in large-scale networks," in *Proc. of ISOC NDSS*, 2014.

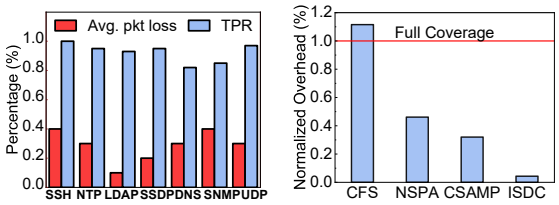
[86] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad, "Webwitness: Investigating, categorizing, and mitigating malware download paths," in *Proc. of USENIX Security*, 2015.

[87] L. Liu, G. Engelen, T. Lynar, D. Essam, and W. Joosen, "Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018," in *Proc. of CNS*, 2022.

Appendix A Other Issues

FI Overhead. For a better understanding of FI's overhead, we extend our FI flow prioritization analyses (see section 6.5.1) and analyze the average packet loss across different attack types, as shown in Figure 16 (a). The results indicate that for most attack types, FI successfully predicts over 96% of flows. However, reflection attack types (e.g., DNS attack) exhibit relatively lower prediction coverage, 87% on average, suggesting a decrease in the attack rate due to flow reflection. On average, attack flows had a 23% packet loss which is inevitable cost for flow size prediction. However, considering the small size of attacks in the dataset (i.e., 34 packets on average), the cost is negligible.

Northbound Overhead. The northbound overhead refers to the amount of data communication towards security applications, essential for collecting and preprocessing data to facilitate the construction of a single model for learning network traffic patterns. NSPA [33] and CSAMP [32] already rely on a central controller for data collection and decision-making, making it trivial to send the feature data to the collector as well. In contrast, CFS [34] necessitates a central collector to remove the fragmentation effect. On the other hand, ISDC does not require data communication with a northbound server but only local model parameters for model aggregation. With the size of the model we employed in our simulation, ISDC can save out-of-band (northbound) overhead up to 94% compared to CFS, as shown in Figure 16 (b). For CFS, the amount of data transmission is even higher than the full population flow feature migration to the northbound server due to the massive amount of fragmentation generated during the flow feature measurement process.



(a) FI Overhead (b) Northbound Overhead

Figure 16: Analyses of potential overheads of ISDC.

Appendix B Dataset Description

We used three datasets from CIC for evaluations [62–64]. To preprocess and extract attack flows, we adopted Liu et al.'s approach [87] for dataset sanitization. In our endeavor to select attack traffic from various attack types, we encountered three challenges that necessitated excluding certain attack types from our evaluation. Firstly, sparse flows of layer-7 attacks, like Slowloris, exhibit a slow nature. Secondly, several attack types had an inadequate number of flows (less than 1k) or were small in size, with 5-tuple definition for meaningful assessment. Lastly, certain attack traffic posed difficulty differentiating from benign traffic when utilized in machine learning models for training purposes. Given that the feature and machine learning are not the focus of this work, we opted to discard these attacks. After the aforementioned preprocessing procedure, the selected attacks fall into three groups, namely reflection (NTP, LDAP, SSDP, DNS, SNMP), brute-force (SSH), and flooding (UDP, UDP-Lag). Table 5 shows the number of flows selected for our evaluation.

Attack type	# of packets	# of flows	Trace
LDAP	5.4M	2K	CIC-DDoS2019 [64]
DNS	17M	161K	CIC-DDoS2019 [64]
NTP	10M	131K	CIC-DDoS2019 [64]
SSDP	1.3M	66K	CIC-DDoS2019 [64]
SNMP	7.3M	28K	CIC-DDoS2019 [64]
UDP-Lag	74K	2.5K	CIC-DDoS2019 [64]
UDP	1.7M	86K	CIC-DDoS2019 [64]
SSH	163K	5.9K	CIC-IDS2017 [62]
SSH	4.3M	188K	CSE-CIC-IDS2018 [63]
Benign	250M	14.5M	CAIDA [50]

Table 5: Dataset used in the evaluation.