Cyber-AnDe: Cybersecurity Framework with Adaptive Distributed Sampling for Anomaly Detection on SDNs

Nadia Niknami*, Avinash Srinivasan†, and Jie Wu*
*Center of Networked Computing, Temple University, USA
†Department of Cyber Science, United States Naval Academy, Annapolis, MD, USA

***Abstract*—By decoupling the control plane and data plane in the software-defined network (SDN), the controller gains a comprehensive global view of the network. The SDN controller samples traffic from all switches to effectively manage data plane traffic. The flow traffic sampling rate significantly impacts the accuracy of controller decisions. Although increasing the sampling rate is desirable for improved detection accuracy, it also increases resource consumption on both switches and the controller. Hence, it is crucial to carefully manage the sampling on switches to fine-tune the accuracy of anomaly detection.Existing flow sampling solutions often struggle to strike a balance between detection accuracy, sampling rate, and overhead. To address this challenge, we propose a robust cybersecurity framework for anomaly detection on SDNs through traffic flow inspection. Our proposed framework, Cyber-AnDe, integrates adaptive distributed sampling (ADS) with a Reinforcement Learning (RL) agent to enhance anomaly detection accuracy while minimizing the increase in controller overhead. In our framework, the controller leverages information gathered from each sampled traffic flow to determine whether the flow's state is *malicious*, *suspicious*, or *benign* based on underlying anomaly detection algorithms. Once the flow state is determined, the controller takes the appropriate action with the help of the RL agent. Through extensive simulations and SDN test bed experiments, we confirm a significant improvement of up to 93% in the detection of anomalies based on network traffic compared to existing solutions.**

***Index Terms*—Adaptive sampling, anomaly detection, attack, cybersecurity, intrusion detection, load balancing, network monitoring, sampling rate, software-defined networks.**

I. Introduction

THE software-defined network (SDN) is an emerging network architecture that decouples the network control plane from the data plane [1]. In an SDN, a centralized controller is responsible for all network control decisions. Fig. 1 illustrates an SDN's application layer, control plane, and data plane. The controller monitors all switches in the data plane, and network applications help the controller with traffic management [2]. The deployment of efficient monitoring tools to assist with network management tasks without interfering with normal operations has become increasingly important in modern networks with high traffic volumes. A network intrusion detection system (IDS), which monitors network behavior and inspects data packets for evidence of malicious activity, is a popular and widely used perimeter security appliance to enhance network security, particularly against external attacks.

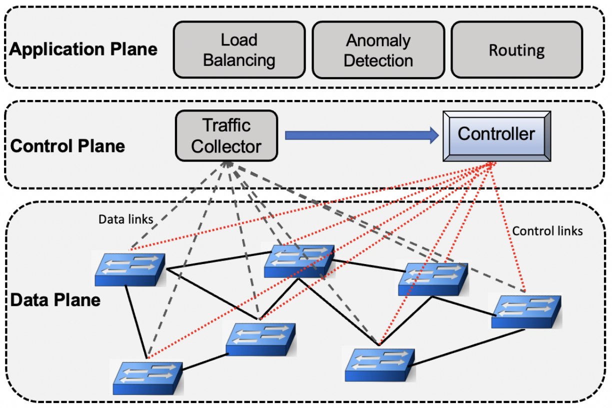


Fig. 1: SDN with sampling-based anomaly detection.

For network monitoring, if every packet in a traffic flow was captured and forwarded to a traffic analyzing engine, then we could potentially have very robust network security. However, such collection and processing of every packet would double the traffic volume with the network soon beginning to experience significant congestion. If the network traffic to be inspected is much larger than the IDS capacity, then the IDS cannot inspect all packets in the network. The SDN controller seeks to minimize unnecessary overhead from the large volume of sampled data sent to the IDS for analysis. Consequently, selectively capturing network traffic through a process known as *traffic sampling* is highly desirable. Network traffic sampling is a plausible approach to make intrusion detection much more efficient by directing only sampled traffic to the controller for analysis. Sampling techniques are vital to achieving efficient network measurements by reducing the amount of traffic that is processed while attempting to maintain the accuracy of network statistical behavior estimations.

Several techniques exist to implement the packet sampling process that can be categorized into two broad types — *static sampling* and *dynamic sampling*. A static sampling process is conducted periodically or randomly following a given rule, which could be *count-based*, *time-based*, or *content-based* rule. Dynamic or adaptive sampling algorithms use different sampling intervals and or rules for data sampling decisions.

This research was supported in part by NSF grants CNS 2214940, CPS 2128378, CNS 2107014, CNS 2150152, CNS 1824440, and CNS 1828363.

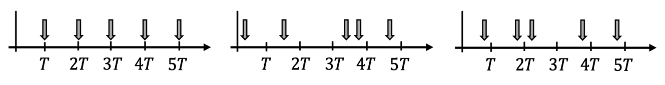
However, these methods overlook the challenge of balancing the trade-off between detection accuracy and the overhead associated with processing the immense volume of traffic. When the appropriate sampling method is selected and strategically implemented, it offers an excellent network view and high detection accuracy, at an acceptable overhead. Current network sampling techniques, primarily designed for traffic engineering purposes, are ineffective for intrusion detection applications. Even with sampling, these methods still incur significant network overhead.

In this paper, we introduce Cyber-AnDe, a cybersecurity framework with an adaptive distributed sampling (ADS) method for SDN traffic monitoring to address the aforementioned limitations. Our approach starts with a low sampling rate and continuously adjusts it based on the current level of network traffic maliciousness. By beginning with a conservative sampling rate and incrementally increasing it only when benign flows exhibit suspicious behavior, we mitigate the risk of overloading the controller with unnecessary data processing tasks. This adaptive strategy ensures accurate anomaly detection while optimizing resource utilization within the SDN infrastructure. Consequently, the sampling rate's impact on IDS detection in SDN becomes a strategic consideration for achieving both security and operational efficiency.

We employ a Reinforcement Learning (RL) agent within a closed network control loopto dynamically adjust the sampling rate. This control loop comprises essential components and control functions. The intrusion detection system report provides the RL agent with insights into the current network status, classifying traffic as malicious, suspicious, or benign. Based on this status, the RL agent decides on the appropriate actions, particularly regarding the sampling rate on specific switches. By aiding the controller in managing traffic monitoring on network switches, the RL agent enhances the network's adaptability and responsiveness to evolving threat landscapes.

Key contributions of our research presented in this paper can be summarized as follows:

1) We propose Cyber-AnDe, a robust cybersecurity framework for SDN traffic monitoring that effectively and efficiently detects an anomaly. Our framework achieves superior anomaly detection accuracy using an ADS, which also results in significantly lower communications costs between the controller and the switch(es).
2) We have designed an *RL*-based *network control loop* between the controller and the data plane to effectively manage Cyber-AnDe's ADS on SDN switches. The RL agent helps the controller determine the best action to take for sampling flow f on switch s at rate γ based on the current state of the system.
3) We leverage a *multi-objective optimization* problem-solving approach to improve the trade-off between detection accuracy and the monitoring process to reduce overhead on the controller.
4) We introduce an orchestration of flow monitoring that separates sampling decisions for flows based on their state. This orchestration also relieves the controller from managing flow sampling across multiple switches, allowing it to focus primarily on optimizing sampling decisions to enhance anomaly detection accuracy on the data plane.



(a) Systematic (b) Random (c) Stratified random

Fig. 2: Sampling methods.

II. PRELIMINARIES

A. Intrusion detection system in SDN

SDN traffic management can be defined as network traffic monitoring and analyzing measures to improve network performance and quality of service metrics. SDN traffic management techniques can be divided into four distinct categories: *routing*, *load balancing*, *congestion control*, and *flow control*. In SDN, the centralized controller communicates with switches through a protocol like OpenFlow [3] and abstracts the data plane's routing and forwarding with a *match-action* table. A key advantage of SDN centralization is its potential for implementing a machine learning-based IDS solution [4]. The centralization of the controller allows for training a machine learning IDS model to effectively identify and report intrusion events to the controller [5]. While separating the data and control planes in SDN offers a manageable and programmable network, it also expands the attack surface, thereby increasing security concerns. [6].

In this paper, we assume that the controller within the SDN architecture is trustworthy and securely managed. Although we recognize the potential risks associated with a centralized controller, such as the possibility of a single point of failure, addressing controller security is beyond the scope of this paper. Our primary focus is exploring adaptive sampling strategies for anomaly detection within this presumed secure framework. For interested readers, existing solutions including [7], [8] address attacks on SND controller.

B. Sampling methods

With the ever-increasing network scale and traffic volume, it becomes increasingly important to consider the limitations of the IDS in terms of *CPU power*, *memory access speeds*, and *storage capacity*. Traditionally, network management has utilized simple and non-adaptive sampling methods. Such methods use a fixed rule to determine when to sample data on each switch, with the sampling rule itself being either deterministic or random [9]. Fig. 2 presents three main packet sampling methods: *systematic sampling*, *random sampling*, and *stratified random sampling*. The *systematic sampling* method samples data at a fixed time interval with a period of T seconds. The *random sampling* method employs a random distribution function to determine the sample.The *stratified random sampling* method combines random sampling with the fixed-time interval that is used in systematic. While stratified sampling has some benefits, it also introduces the question of how to stratify a population that can create more risk of bias.

When a large-scale anomaly occurs, the possibility of discarding the attack packets increases. If some attack packets

TABLE I: Main Notations.

Symbol	Meaning
f/F	flow/set of flow f
s/S	switch/switch set
p_f	probability of capture failure for flow f
C/C_s	capacity/capacity of switch s
$p_{f,s}$	probability of capture failure for flow f on switch s
r_s	rate of traffic on switch s
m_f	rate of malicious flow f
τ	threshold for determining the level of suspicion of flow
p	probability output of the classifier
γ_s/γ_f	sample rate on switch s / flow f
f^γ	sampled flow
f_N^γ/f_S^γ	normal/strange sampled flow
s_f	list of switches that flow f passes through
μ	capacity of IDS
T_α	time interval between samplings
$U(s,f,\gamma)$	utility value for sampling flow f in switch s using rate γ
$\mathcal{R}_P(t)$	rate of received traffic at time t
$\mathcal{L}_P(t)$	rate of loss traffic at time t
$T_P(t)$	rate of transmitted traffic at time t

are to be discarded, the loss will influence the detection rate greatly. The large-scale anomaly bears some distinct traffic features. For example, the volume of packets and flows will soar and exhaust IDS soon. A sampling rate management application should be capable of controlling the sampling rate to provide sufficient accuracy at minimal overhead for the controller and switches. Every switch may have the same sampling rate for each flow denoted as $\frac{Capacity}{\#of flow}$ or may have a rate that is proportional to its traffic rate denoted as $\frac{Capacity}{\sum_{s\in S} r_s}$.

The sampling rate within SDN switches significantly impacts IDS detection in SDN by directly affecting anomaly detection accuracy and resource utilization efficiency. Ideally, IDSs would monitor all network traffic for optimal performance, but the controller, responsible for managing various control applications, may struggle to analyze large volumes of data, especially during high traffic periods. If network traffic exceeds the IDS capacity, not all packets can be inspected, necessitating the use of SDN functionalities to sample a portion of data traffic from network switches and forward it to the IDS. A higher sampling rate increases the likelihood of capturing anomalous behavior, thus improving detection capabilities but potentially raising resource overhead. Conversely, a lower sampling rate will reduce resource utilization but may result in missed anomalies, thereby compromising network security. Therefore, achieving the optimal balance in sampling rate is crucial for effective IDS detection in SDN.

To illustrate this impact quantitatively, we consider two hypothetical scenarios with sampling rates set at 1% and 5%, both using the same IDS algorithm. In the first scenario, with a 1% sampling rate, the IDS inspects 1 out of every 100 packets traversing the network. This conservative sampling rate reduces the computational burden on both the IDS and the controller but increases the risk of missing certain anomalies, particularly those occurring in less frequently sampled packets. For instance, let us assume that in this scenario, the IDS detects 80% of anomalies present in the network. In contrast, in the second scenario with a 5% sampling rate, the IDS examines 1 out of every 20 packets. The increased sampling rate improves the likelihood of detecting anomalies across a broader spectrum of network traffic but also escalates the computational demands on both the IDS and the controller. For demonstration purposes, let us assume that this increased coverage allows the IDS to achieve a detection rate of 95%. These scenarios highlight the trade-off between sampling rate and IDS detection efficacy. A lower sampling rate conserves network resources but compromises detection accuracy, while a higher sampling rate enhances accuracy at the cost of increased resource consumption. Therefore, determining the optimal sampling rate requires carefully balancing the desired level of anomaly detection with efficient network resource utilization.

III. The Proposed Cyber-AnDe Framework

In this paper, we propose Cyber-AnDe, a cybersecurity framework for SDN network traffic flow inspection with a robust ADS method. The framework has an SDN-based unified architecture composed of an SDN controller, OpenFlow-enabled switches, and necessary custom implemented applications, as shown in Fig. 1. Our framework runs the ADS on each switch to enhance the inspection performance, while keeping the total aggregated sampled traffic below the capacity of the traffic collector. Our framework also minimizes the overhead on both the controller and the individual switches to avoid potential conflicts between data packets and control packets. The controller is responsible for analyzing the sampled traffic to detect anomalies.

In our framework, the controller combines the OpenFlow and sFlow [10] protocols. OpenFlow is a programmable networking protocol that enables the controller to interact with the switches. The OpenFlow-enabled switches primarily serve as forwarding elements. The proposed Cyber-AnDe framework, presented in Fig. 3, includes the following key components:

1) *Traffic Sample Repository (TSR):* This module collects the sampled traffic flows from the sampling switches of the data plane.
2) *Behavior Monitor Application (BMA):* This module is responsible for checking the sampled traffic's fields and identifying the headers. BMA can easily observe the packet's structure. It can roughly estimate the flow number and aggregate statistics, which can be helpful to detect anomalies.
3) *Sampler Scheduler Application (SSA):* This module determines the sampling strategy, i.e., which flow should be sampled by which switch and at what rate.

The Traffic Sampling and Reporting (TSR) module is responsible for collecting sampled traffic from switches in the data plane and is a sub-component of Cyber-AnDe in the control plane. If the TSR module is integrated within the SDN switch, it can have both positive and negative impacts on the switch's security and performance. On the positive side, processing sampled traffic locally within the switch reduces the latency associated with sending traffic data to a centralized controller for analysis. This local processing can enhance the network's responsiveness to security events. However, integrating the TSR module within the SDN switch introduces

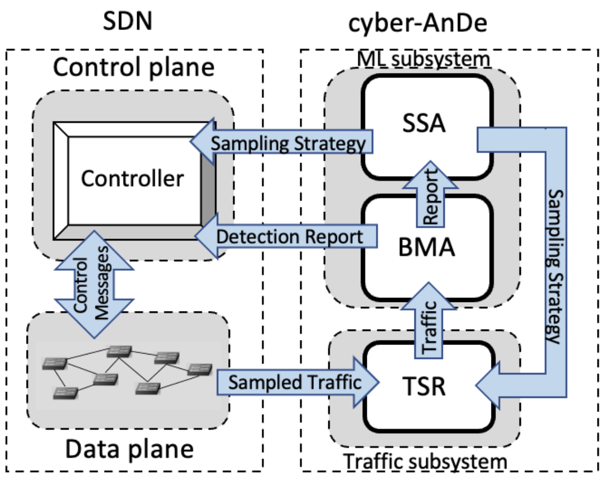


Fig. 3: Control plane, data plane, and Cyber-AnDe.

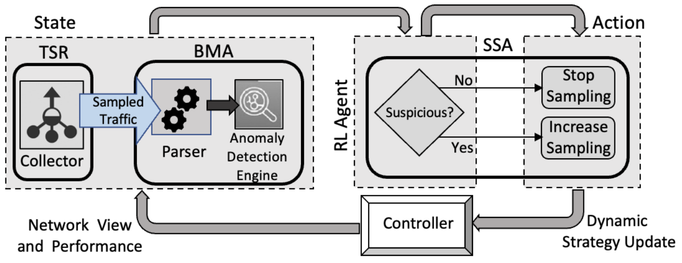


Fig. 4: The proposed Cyber-AnDe framework.

additional processing overhead. This increased load can potentially affect the switch's performance, particularly if it is already operating near its capacity. The TSR module would also consume scarce switch resources such as CPU cycles and memory, which are essential for packet forwarding. Under heavy traffic loads or in switches with limited resources, this additional resource consumption could lead to performance degradation.

The BMA module processes the sampled traffic it receives from the TSR. During this processing, the BMA module analyzes packet header features such as *source IP*, *destination IP*, *source port*, *destination port*, *transport protocol*, *flow size*, and *packet count fields*. These features directly reflect the state of the traffic. After the analysis, the BMA generates a report detailing the flow statistics and behavior of the sampled traffic, which is then sent to both the controller and the SSA module. This report determines whether the traffic is classified as benign, suspicious, or malicious. We establish a threshold, denoted as τ, to gauge the level of suspicion. This threshold τ defines the confidence level of the predictions made by the attack classifier within the BMA module. Fig. 4 shows the flow of information and control between the BMA and SSA modules within the ML subsystem.

Based on input from the BMA, the SSA module adjusts the sampling rate. While a higher sampling rate is generally preferable, as it enhances the controller's accuracy in detecting malicious traffic, caution is necessary. Blindly increasing the sampling rate can lead to diminishing returns, resulting in network congestion and added overhead on both the sampling switches and the controller. The proposed Cyber-AnDe's ADS method starts at a minimum sampling rate and then gradually increases the sample size until no further improvement, in terms of detection accuracy, can be achieved. SSA also sends an update to the controller recommending what sampling strategies to use on different switches. Based on the input from the SSA and BMA modules, the controller determines the switches and rates at which the flows should be sampled.

In our framework, we maintain two tables for each switch: a *forwarding table* and a *sampling table*. The controller populates and maintains the entries of both these tables. The forwarding table is populated and updated based on the underlying routing algorithm. It has information about forwarding packets from the source to the destination, which is set by the controller. On the other hand, the sampling table is populated and updated based on the SSA module's input. The sampling table has information regarding the sampling rate and the time interval for forwarding sampled traffic to the BMA module.

Fig. 5 shows the timeline for message exchange between the various components of our framework, including the SDN. Updating *sampling strategy messages* and *forwarding messages* are the actions of the controller, which are based on the reports the controller receives from the ML_Subsystem. The FWDApp is the default forwarding application in the SDN, helping the controller install proper flow rules on the flow tables of each switch. There are three possible actions that a controller can choose from:

1) *Blocking:* When the attack traffic information is exact and does not overlap with any non-malicious traffic, the controller installs a block rule on the flow table of the switch associated with the attack traffic.
2) *Rate Control:* When the traffic observed is suspicious, the controller increases the sampling rate for deeper analysis.
3) *Forwarding:* When traffic is non-malicious, the controller finds the forwarding path and installs the flow rule on the flow table.

These actions are configured on switches in the data plane through messages from the controller. We consider different types of control messages that are sent from the controller to the switches in the data plane:

1) *Forward-flow message: (interface, path, action:forward)*
2) *Block-flow message: (interface, src, action:block)*
3) *Sample-flow message*: *(interface, interval, sample size, action:sampling*)

If the sampled traffic is identified as malicious according to the BMA log, the controller should promptly handle the Block-flow message. For suspected traffic, the flow inspection schedule should be adjusted. The BMA's task is to monitor the traffic to detect any inconsistencies in the data plane traffic. In addition, this application checks the frequency of packet-in messages and extracts some features to detect anomalies. Additionally, this application monitors the frequency of packet-in messages and extracts features to detect anomalies. It assigns a security score based on these features within a given time

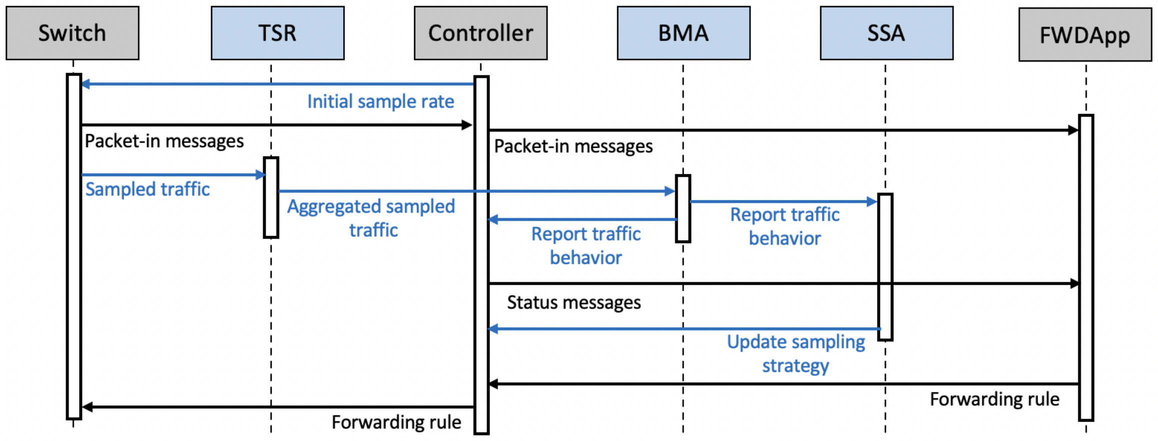


Fig. 5: Control and information flow in Cyber-AnDe.

period (T) and categorizes incoming traffic into three types of flows: newly generated, suspicious, and malicious.

A. *Placement of sampling switches*

Network traffic is monitored by capturing traffic flows at strategic location, which significantly impacts the monitoring performance. There are two possible locations for traffic capture – *core switches* and *edge switches*. Unlike a core switch on the network backbone, the number of flows at an edge switch is relatively small since it is directly connected to client devices. Therefore, it is possible to monitor individual flows that pass through each edge switch at a more granular level. As each monitored edge switch requires a dedicated capturing device, the number of capturing devices required for network-wide traffic monitoring will be substantial. Furthermore, managing many hardware devices will result in substantial additional management overhead. Therefore, this solution approach to network traffic monitoring for anomalies is not a practical solution for large networks.

SDNs do not necessitate differentiating edge and core switches [11]. Leveraging hardware-based capturing devices is important to capturing traffic on SDNs. Such hardware-based capturing further relies on OpenFlow to capture traffic packets. In practice, this involves updating the flow table of each switch using OpenFlow. In the proposed Cyber-AnDe framework, we design a greedy set-cover algorithm that iteratively selects the most cost-effective set and removes the covered elements. The same greedy set-cover algorithm also selects the location of traffic capture by selecting a subset of switches as sampling switches. Once a subset of switches is chosen, packets passing through each switch can be fairly sampled at a certain rate. Our proposed greedy set-cover algorithm guarantees the size of the set sampling switch is minimal. We are cognizant of the capacity limitation of switches and design a separate *capacitated set-cover algorithm* [12] [13].

The capacitated set-cover algorithm takes into consideration the capacity limitation when selecting the switches. The capacitated set-cover switch selection problem can be formulated

Algorithm 1 Capacitated Set-Cover Selection

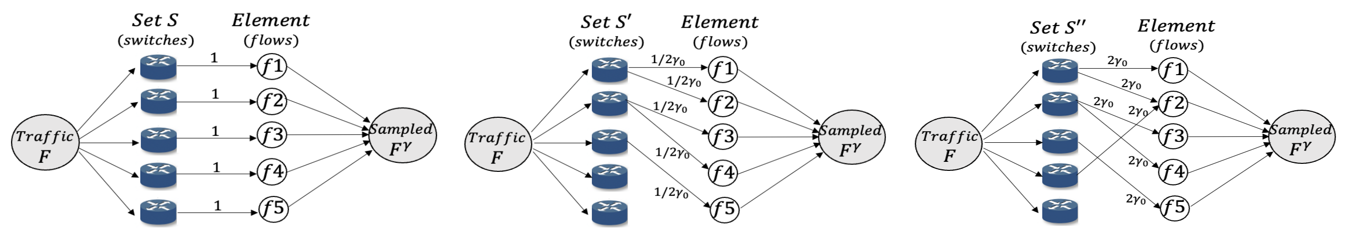
```
Require: Switch set $S$, Flow set $F$, Capacity $C$
 1: Initialize total number of the sampling switch $k$
 2: Initialize cover switch set $\mathcal{S}^s = \varnothing$
 3: Initialize cover flow set $\mathcal{F}^s = \varnothing$
 4: for each switch $s \in (S - \mathcal{S}^s)$ do
 5:     while $C_s > 0$ do
 6:         for each $f \in (F - \mathcal{F}^s)$ do
 7:             Flow $f$ will be sampled by switch $s$
 8:             $\mathcal{S}^s \leftarrow \mathcal{S}^s \cup s$
 9:             $\mathcal{F}^s \leftarrow \mathcal{F}^s \cup f$
10:             Calculate remaining capacity $C_s$
11: return List of sampling switches $S^s$
```

as an optimization problem as follows:

$$\begin{aligned} \textit{minimize:} \quad & \sum_{i=1}^{k} C_i.x_i \\ \textit{subject to:} \quad & \sum_{f \in F} 1/2\gamma_0 * r_f.x_i < C \\ & x_i \in \{0,1\} \quad \forall S_i \in S. \end{aligned}$$

Here, x_i indicates whether switch S_i is chosen or not. r_f denotes the traffic flow rate f, and γ_0 presents the minimum sampling rate. For each switch, the total amount of sampling is at most C. Algorithm 1 shows *Capacitated Set-Cover* selection for the sampling switches. Based on the switches' capacity, C_s, a subset of switches $S^s \in S$ is selected to cover all the incoming traffic. Suppose we change the sampling rate from 1 to minimum sampling rate $1/2\gamma_0$, a subset $S^{s'} \in S^s$ can cover the traffic. Algorithm 2 illustrates the process of selecting $S^{s'}$ from the regular set-cover algorithm (or from the set cover output by Algorithm 1).

A toy example: Fig. 6 shows the steps involved in selecting the sampling switches and setting the sampling rate. S is the set of candidate switches for the sampling based on the capacitated set-cover. As mentioned before, the selection of switches in this step is based on the assumption that each switch can cover one flow. Fig. 6(a) displays the capacitated set-cover S. Now, with the help of a regular set-cover algorithm, a subset of switch $S' \in S$ can be selected. Fig. 6(b) displays the capacitated set-cover S'. The sampling rate for the



(a) Capacitated set-cover (b) Set-cover $S' \in S$ (c) Dynamic set-cover S''

Fig. 6: Placement of sampling switches.

Algorithm 2 Set-Cover Selection

```
Require: Switch set $\mathcal{S}^s$, Flow set $F$, Sampling rate $\gamma_0$
1: Initialize cover switch set $\mathcal{S}^{s\prime} = \varnothing$
2: Initialize cover flow set $\mathcal{F}^{\prime s} = \varnothing$
3: $\gamma = 1/2\gamma_0$
4: for each $f \in (F - \mathcal{F}^{s\prime})$ do
5:     for each $s \in \mathcal{S}^s$ do
6:         if $(r_f \times \gamma) < C_s$ then
7:             Assign flow $f$ to switch $s$ for sampling
8:             $\mathcal{S}^{s\prime} \leftarrow \mathcal{S}^{s\prime} \cup s$
9:             $\mathcal{F}^{s\prime} \leftarrow \mathcal{F}^{s\prime} \cup f$
10:            $C_s = C_s - (r_f \times \gamma)$
11:            break
12: return List of sampling switches $\mathcal{S}^{s\prime}$
```

Algorithm 3 Adaptive Distributed Sampling Algorithm

```
Require: Flow set $F$ and switch set $\mathcal{S}^{s\prime}$
1: Initialize $\gamma = 1/2\gamma_0$
2: $\mathcal{S}^{s\prime} \leftarrow$ Candidate switches selected by Algorithm 2
3: for each switch $s \in \mathcal{S}^{s\prime}$ do
4:     Switch $s$ sends traffic sampled at rate $\gamma$ to TSR
5: TSR gathers sampled traffic
6: TSR sends report to the controller
7: while Not Converged do
8:     for each suspicious flow $f$ do
9:         $\gamma' = 2 \times \gamma$
10:    if switch $s$ can handle $\gamma'$ then
11:        Switch $s$ continues with new sampling rate
12:    else
13:        for each switch $s' \in \mathcal{S}'$ do
14:            $C'_s$= Remaining capacity of $s'$
15:            if $2\gamma' < C'_s$ then
16:                Migrate flow $f$ to the switch $s'$
17:            else
18:                Migrate flow $f$ to the new switch $s_m$
19:                Allocate remaining sampling rate to $s_m$
```

switches in S' is based on the minimum sampling rate, which is $1/2\gamma_0$, which is less than the sampling rate associated with the selected switches in S. Therefore, this minimum number of sampling switches can cover all the flows.

When a suspicious flow is detected, the controller must continue increasing the sampling rate on the affected switch until it reaches the predefined confidence level. This confidence level enables the controller to determine whether to forward the sampled traffic as legitimate or block it as malicious. However, switches have a limitation on how much sampling they can handle. When a switch reaches its maximum capacity for sampling rates, it becomes necessary to migrate the sampling of suspicious flows to other switches. In such cases, the controller reallocates a portion of the sampling load to a new switch. Fig. 6(c) illustrates the reassigning of the sampling task of suspicious flow f_2. By doing so, the traffic collector gathers sufficient sampled traffic to analyze the suspicious flows effectively. This enables the controller to determine the appropriate action for the incoming traffic.

Theorem 1. Let $CSC(F, S, C)$ be a capacitated set-cover problem instance. Capacitated set-cover S covers all flows $f \in F$, where it is subject to the sampling capacity C. If we change the sampling rate from 1 to the minimum sampling rate $1/2\gamma_0$, $CS(F, S, C, \gamma_0)$ selects a subset S' from S with the help of the regular set-cover problem, and the max number of flows cover by each sampling switch in set S'.

Proof: Suppose that each flow f needs a sampling rate of 1. The capacitated set-cover problem selects some switches covering all the flows f under capacity C. By changing sampling rate from 1 to $1/2\gamma_0$, each switch can cover more flow because $1 \times r_f > \gamma_0 \times r_f$ when $\gamma_0 < 1$. Therefore, by using the regular set-cover on S, we have the minimum number of sampling switch set S' that covers all the flows f. ■

B. Network control loop

The proposed Cyber-AnDe framework utilizes a closed network control loop (including an RL agent) to dynamically adjust the sampling rate. The control loop is composed of necessary components and control functions, as illustrated in Fig. 4. Generally, the traffic collector starts with an initial sampling rate, γ_0, which is distributed among the selected switches. The controller in Cyber-AnDe operates as part of a closed control loop to determine the optimal sampling rate and the best action for each incoming network traffic flow. This closed control loop allows the sampling rate to be automatically adjusted based on the state of the traffic flow as assessed by the controller. Details of the sampling rate adjustment process are outlined in Algorithm 3. The anomaly detector sends a report to the controller, which then decides on the subsequent sampling process based on this report and its learning from the current status of the data plane.

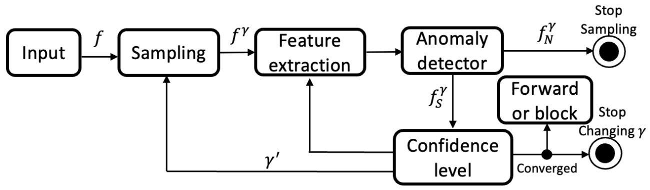


Fig. 7: Closed control loop of Cyber-AnDe framework.

For suspicious traffic, the controller increases the sampling rate until reaching a convergence point regarding the status of the given traffic. However, switches have limited capacity and may be unable to handle the updated sampling rate. To make an accurate decision, the switch will require assistance processing at the new sampling rate. Consequently, the controller will add a new switch to the sampling set. The dynamic set-cover method [14], outlined in lines 13-20 of algorithm 3, effectively addresses this issue.

As discussed, increasing the sampling rate results in more sampled traffic, which, when forwarded to the traffic collector, consumes network resources and may cause congestion, interfering with normal traffic flow. Therefore, it is crucial to minimize duplicated packets for traffic sampling to reduce the impact of traffic monitoring. The detection engine, used for analyzing sampled traffic during inspection, has limited processing capacity. If the incoming traffic rate exceeds its capability, the detection engine will drop packets. Thus, sampled traffic should not exceed the detection engine's processing capacity.

Fig. 7 illustrates the RL control loop and stopping points. RL is a sub-field of machine learning that addresses the problem of learning optimal decisions over time. It is a machine learning technique based on the Markov Decision Process (MDP) and can be used to tackle this task. In RL, the agent keeps interacting with the environment to find the optimal policy π to maximize his expected accumulated rewards [15]. RL aims to learn a policy to determine which action a to take given a specific environment represented by the state s. In our situation, the state s can be presented by one of three possible traffic statuses on each switch: 1) Legitimate, 2) Suspicious, and 3) Malicious.

The flow's status is determined by evaluating the probability outputs of the classifier in BMA, where p_1 is the probability of being malicious and p_2 is the probability of being benign. If $(p_1 > \tau)$, the flow is categorized as malicious. If $(p_2 > \tau)$, the flow is categorized as benign. Conversely, if neither p_1 nor p_2 exceeds τ, the flow is flagged as suspicious. This threshold-based classification mechanism enables the identification of potentially anomalous network behavior for further assessment.

Based on the status that RL determines, the corresponding action a to take can be one of:

1) Increasing the sampling rate,
2) Stopping the sampling rate,
3) Adding an assistant switch.

In RL, the reward reflects the success of the agent's recent activity and not all the successes achieved by the agent so far. In our approach, the agent's objective is to learn the policy of monitoring the traffic that maximizes the expected detection rate and guarantees minimum overhead for the controller. The Q function for reward is defined as follows:

$$\begin{aligned} Q_{t+1}(s_t, a_t) =& (1-\alpha)Q_t(s_t, a_t) + \alpha(U(s_t, a_t) \\ &+ \lambda \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})), \end{aligned} \tag{1}$$

where $U(s_t, a_t)$ denotes the reward value of a given action a_t at state s_t.

RL would be helpful based on the traffic status on the given switch, which is based on the average rate of received traffic and the average rate of loss traffic. We formulate average rate of traffic $\overline{\mathcal{R}}(t)$ as $(R_p(t) - R_p(t-\tau))/\ (t-\tau)$, where $R_p(t)$ is received traffic rate at time t and τ denotes the end of the previous time interval. The average loss of traffic $\overline{\mathcal{L}}(t)$ can be formulated as $(R_p(t-\tau) - T_p(t-\tau))/\ R_p(t-\tau)$, where $T_p(t)$ represents the transmission rate of traffic at time t. The observed state of a switch is denoted as follows: $state = \{(\overline{\mathcal{R}}_1(t), \overline{\mathcal{L}}_1(t)), ...(\overline{\mathcal{R}}_n(t), \overline{\mathcal{L}}_n(t))\}$. We define actions of reallocation as: 1) New allocation (extra space) for control at time t and 2) New allocation (extra space) for data at time t. Here, RL aims to minimize the penalty, which is the cost of lack of space. The Q function is defined as:

$$\begin{aligned} Q_{t+1}(s_t, a_t) =& (1-\alpha)(Q_t(s_t, a_t) + \alpha(P(s_t, a_t) \\ &+ \lambda \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})), \end{aligned} \tag{2}$$

where $P(s_t, a_t)$ is the penalty function, and the value of penalty can be calculated by $C \times R_p(t)/b_p(t)$ where $R_p(t)$ and $b_p(t)$ represent the current rate and the allocated rate respectively. Minimizing the probability of capture failure is the objective of the sampling methods. We can formulate the objective as $\min_\gamma\{\max_f p_f\} = \min_\gamma\{\max_f \prod_s p_{f,s}\}$.

We need to evaluate the framework's performance with different sampling rates resulting from applications. We define a utility function $U_f(s, f, \gamma)$ to find the best option to the current system for sampling flow f in switch s using rate γ. The utility function can be defined as follows:

$$\begin{aligned} U_f(s, f, \gamma) =& \sum\nolimits_{s \in S} \sum\nolimits_{f \in F} (\alpha \cdot \mathcal{G}(f_s, r_s, \gamma_f) \\ &- \beta \cdot \mathcal{M}(f) - \zeta \cdot \mathcal{P}(T, \tau)), \end{aligned} \tag{3}$$

where $\mathcal{G}(f_s, r_f, \gamma_f)$ denotes the function computing the accuracy of detection on flow f_s on the given switch s with data rate r_f and sampling rate γ_f. The function $\mathcal{M}$ represents the cost value for computation/processing of flow f and communication between components. The function $\mathcal{P}$ represents the penalty, which is based on the delay in this approach. This utility function helps the controller to find [switch, flow, rate] by considering the capacity limitation of the TSR. We can formulate this problem as an optimization problem:

$$\begin{aligned} &\textit{maximize:} && U_f(s, f, \gamma) \\ &\textit{subject to:} && \sum\nolimits_{f \in F} r_f \cdot \gamma_s \le C \quad \text{for each } s \in S \\ &&& \sum\nolimits_{s \in S} s_f \ge 1 \quad \text{for each } f \in F. \end{aligned} \tag{4}$$

The total sampling rate is calculated by summation of flows' sampling rates, denoted by γ_s for each switch $s \in S$. The aggregated volume of sampled packets is retained below the maximum TSR capacity. The total capacity of the TSR is denoted by C in bits per second. Let s_f denotes the list of switches that flow f is sampled on. Each flow f should be sampled at least one switch. The controller updates the time interval between the sampling based on the current state of the network. T_α representing the time interval α. Sampling begins with a low period T_0. The controller will update the time interval according to $T_\alpha = (1+\alpha)T_0$, where α is a confidence index. The controller can change α based on the result of the detection. If there is no alert from the TSR, the controller will increase α. Otherwise, it will be decreased. We need to take into account α_{max} to control the maximum time interval between samplings. Therefore, we consider $\alpha = \min(\alpha+1, \alpha_{max})$ in the implementation of our adaptive sampling module in the Cyber-AnDe framework. Sampling generally takes place for a predetermined period of time. In contrast, duration needs to be carefully determined as each security attack requires a different observation time to be accurately classified with a high detection rate and a low false positive rate.

A *false negative* is an important metric for intrusion detection. It shows the rate of not detecting an attack when an attack has taken place. If the malicious traffic is not sampled, then the intrusion detection component does not detect the existence of the attack. Therefore, different sampling strategies should be applied in various cases, and we will discuss them separately in the following sections. The performance of the intrusion detection component can be shown by the *false negative* rate. We define the *false negative* rate as follows:

Definition 1. (False Negative Rate) A flow f passes several switches to reach its destination. Let p_f be the probability that the BMA fails to capture flow f^{th} as $p = p_1, ..., p_f$, where $0 \le p_f \le 1$. The total probability of capture failure for the f^{th} flow is the product of the probability of capture failure at every passed switch denoted by $p_f = \prod_{s \in S} p_{f,s}$, where $p_{f,s}$ is the probability of capture failure for f^{th} flow in a switch s. The process of sampling packets is the same as picking $r_s.\gamma_s$ balls out of r_s balls. Therefore, the probability of capture failure for flow f on switch s is $p_{f,s} = \binom{r_s - m_f}{\gamma_s \cdot r_s} / \binom{r_s}{\gamma_s \cdot r_s}$. We can rewrite as $(r_s - m_f)! \cdot (r_s - \gamma_s \cdot r_s)! / r_s! \cdot (r_s - m_f - \gamma_s \cdot r_s)!$.

C. Detecting the Convergence

A key assumption behind Cyber-AnDe's ADS is that convergence can be detected accurately and efficiently. The Cyber-AnDe's ADS method continues taking samples according to the report received from SSA until it detects convergence. The convergence point is still unknown or is an open problem [16]. We need some bound for termination. We check the accurate measurement of the algorithm for where the controller can get a good approximation. Within the scope of our research presented in this paper, we can define a bound for the anomaly detector. Whenever the detector reaches this bound, the controller does not need to continue updating the sampling rate. The value for this bound can be set by entropy threshold and ML classification methods. Convergence detection is fundamentally a statistical judgment to estimate whether the given criterion has been met. Specifically, convergence is reached when $Pr((U(N) - U(n_i)) > \epsilon) \le \delta$, where $U()$ is the utility of the model, ϵ denotes the maximum acceptable decrease in utility, and δ is a probability that the maximum utility value difference from the previous run will be exceeded on any run.

IV. Related Works

A. Sampling Strategy for Detection of Malicious Traffic

Previous studies have dived into the impact of sampling on network management and anomaly detection [17]–[20]. Campazas-Vega *et al.* [21] presented influence of sampling rate in different detection methods based on machine learning. They trained machine learning-based models with different flow datasets. They found out that with a sampling threshold of 1 out of 250 packets, all the algorithms tested achieved high detection rates. Also, they proved that the KNN-based one is the best model in terms of detection capability at all thresholds, but also the worst in execution performance. Du *et al.* in [22] introduced self-adaptive sampling, tailoring the sampling probability to each flow's size and spread. Biswas in [23] claimed the effect of sampling rate on the DDoS detection rate of the monitors. They proposed a flow grouping approach based on behavioral similarity among virtual machines. They formulated the problem as an optimization problem and found the optimal sampling rate distribution.

B. Sampling Strategies in SDN

In the world of SDN network architecture, researchers have been experimenting with the fusion of IDS with SDN [24]–[28]. Kim *et al.* in [29] proposed a less-intrusive traffic sampling mechanism for multiple traffic analyzers on an SDN-capable network using a deep deterministic policy gradient (DDPG), which is a representative deep reinforcement learning algorithm for continuous action control. Wang *et al.* in [30] proposed a spatial-temporal collaborative sampling framework, with the aim of maximizing the sampling accuracy for different flows considering the influences of switches and the collaborative strategy among switches in the spatial-temporal dimension. Their proposed approach improved the sampling accuracy of total/mice/elephant flows and reducing the redundant sampling packets, which can be applied in the practical systems.

Ujjan *et al.* in [10] introduced sFlow and adaptive polling-based sampling with an IDS deep learning model to detect unwanted IoT nodes DDoS traffic at data plane of SDN. Sadrhaghighi *et al.* in [31] presented a per-flow sampling system, optimizing for long flows by delegating short-flow sampling decisions to edge switches. Cai *et al.* in [32] proposed a spatial temporal collaborative sampling (STCS) for SDN. They considered the influences of nodes and the effect on sampling accuracy imposed by the collaborative strategy among nodes in the time dimension.

In contrast to previous works, our proposed method, Cyber-AnDe's ADS, leverages central control in SDN and feedback from the intrusion detection loop. The controller dynamically adjusts the sampling rate based on traffic sampling repository data and behavior monitoring. In cases where the switches reach their sampling capacity, the controller smoothly transitions to new sampling switches using a migration method, providing an adaptive and efficient sampling solution.

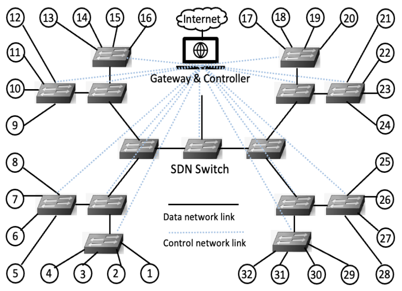


Fig. 8: SDN topology.

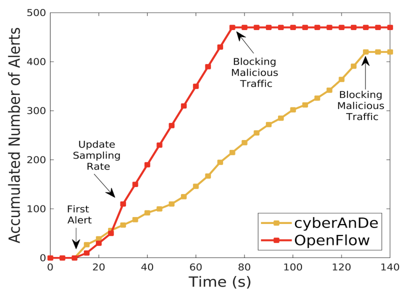


Fig. 9: Number of alerts.

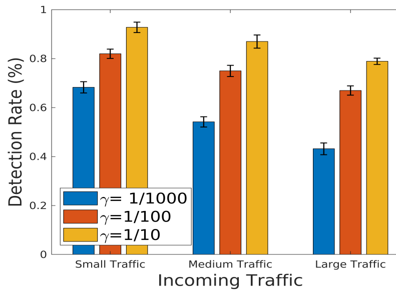


Fig. 10: Effect of γ on IDS.

V. Performance Evaluation

In this section, we present a comprehensive evaluation of the CyberAnDe approach, focusing on its performance across various scale of traffic, malicious traffic rate, and network topologies. The primary objectives of this evaluation are to assess the detection rate and overhead associated with Cyber-AnDe. To evaluate the proposed approach, we provide traffic including both malicious and normal flows. For generating malicious traffic, we utilize the *hping3* tool, with which we can launch *SYN flood*, *UDP flood*, and *malformed packet* attacks at different data rates. The legitimate traffic is generated using the *Ostinato* traffic generator, which can operate in one of two modes – *normal mode* or *burst mode*. Finally, the *ntopng* tool provides a detailed view of the sampled traffic. We have developed and implemented an SDN application to collect sampled traffic using *Snort* in *signature* mode.

Our data-center, shown in Fig. 8, is composed of the following – 35 servers, 15 SDN switches, and 4 regular L2 switches. All the servers, except the Gateway, are Dell PowerEdge 210 servers with the following configuration - 2 cores 2.4 GHz processor, 4 GB RAM, 500 GB storage, minimum 2 gigabit Ethernet ports. We set up two networks – a *control network* and a *data network*. An L2 switch connects all management ports of SDN switches and the SDN controller in the control network. SDN switches are configured as out-of-band controllers. The data ports of the SDN switches and the Gateway are connected in the data network resulting in a three-level complete binary tree topology. The Gateway is connected to the root SDN switch, and the other servers are connected to the leaf SDN switches. We use *Open Network Operating System* (ONOS) as the SDN controller, which is installed on the Gateway. Using the *OpenFlow Discovery Protocol*, ONOS communicates with OpenFlow switches and maintains a state graph of the switches. Additionally, it exposes a northbound API to the OpenFlow applications. We install routing flow rules using ONOS proactive forwarding.

We consider the following four switch selection algorithms to evaluate our proposed method against different approaches:

1) *Top-k based on Betweenness Centrality (BC)*: select the K sampling switches with the highest structural influence in the topology. [33],
2) *Random-k*: select K switches randomly,
3) *Set-Cover selection (SC)*: select K switches maximizing the coverage of all current active flows,
4) *Adaptive Distributed Sampling (ADS)*: switch selection algorithm in Cyber-AnDe.

A. Measurements

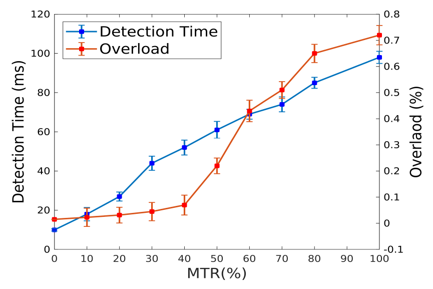
Increasing sampling rates results in an increase in the amount of traffic that is sampled. Due to the fact that the sampled traffic is forwarded to the TSR, it consumes network resources and may lead to network congestion, which interferes with the normal collection of traffic flow data. To evaluate our approach, we report the following performance metrics:

- **False Negative** – percentage of malicious packets that were not detected. This is because of a lack of enough samples to detect malicious status for a given flow.
- **Detection Rate** – the proportion of the whole sample where the anomalies were detected correctly.
- **Detection/Responding Time** – the duration between the occurrence of a security event (such as an intrusion or anomaly) and the time the IDS identifies, send report to the controller and the controller responds to it.
- **Network Sampling Load** – the number of control messages sent from the controller to the switches.
- **Minimum Sampling Switches** – absolute minimum number of switches required to provide the necessary traffic flow coverage for sampling that achieves the required anomaly detection rate.

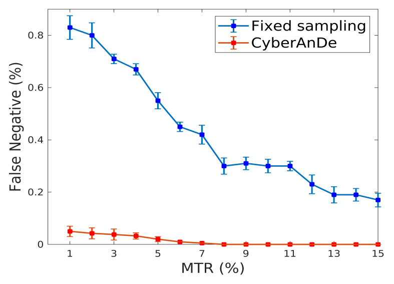
B. Evaluation Results

Fig. 9 illustrates the performance of our proposed Cyber-AnDe framework with early detection and early response. Compared to OpenFlow, Cyber-AnDe achieves approximately 46% faster detection at the cost of approximately 15% increase in number of alert messages as overhead for the controller. To find the relationship between the sampling rate and the detection rate, we need to vary the sampling rate and record the detection rate.

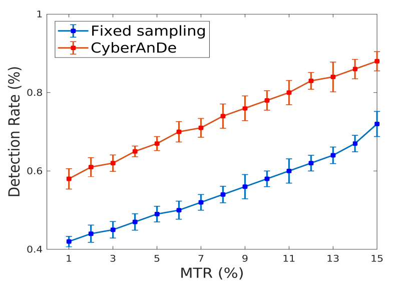
Fig. 10 displays how the sampling rate in SDN switches impacts the IDS detection in SDN. Three different sampling rates – $\gamma_0 = 1/10$, $\gamma_0 = 1/100$, and $\gamma_0 = 1/1000$ – are compared across three different traffic scales – *small*, *medium* and *large* with $MTR = 2.5\%$ and *switch capacity* $= 1Gb/s$. This allows us to explore the effect of sampling rate on detection performance. The analysis of the results reveals a clear trend: higher sampling rates ($\gamma_0 = 1/10$) consistently yield better detection rates across all traffic levels. For small, medium,



(a) Time vs. overhead - switch capacity: 1Gb/s

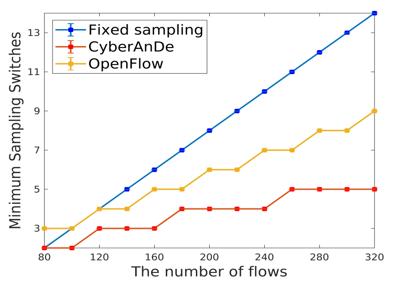


(b) False Negative - switch capacity: 1Gb/s

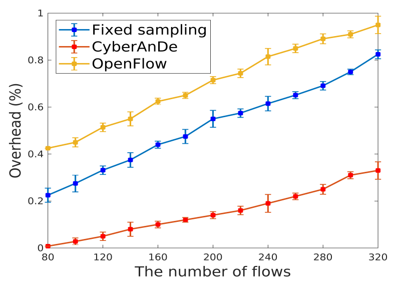


(c) Detection Rate - switch capacity: 1Gb/s

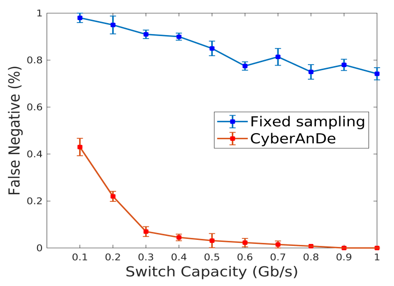
Fig. 11: Impact of MTR on various performance metrics when $\gamma_0 = 1/100$.



(a) MTR: 2.5% and detection accuracy:90%



(b) Controller's overhead - MTR: 2.5%



(c) False Negative - MTR: 2.5%

Fig. 12: Impact of the scale of incoming traffic when $\gamma_0 = 1/100$.

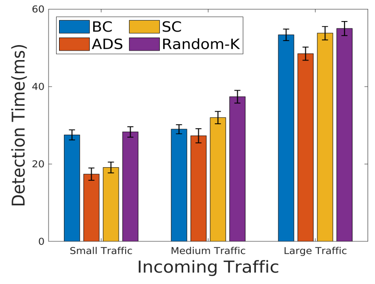
and large traffic volumes, the detection rates are highest with $\gamma_0 = 1/10$, followed by $\gamma_0 = 1/100$ and $\gamma_0 = 1/1000$. This trend suggests that increasing the sampling rate enhances the system's ability to detect intrusions effectively. Therefore, optimizing the sampling rate to higher values is crucial for improving the IDS's detection performance, especially as incoming traffic volume increases.

In Fig. 11, we present Cyber-AnDe's detection time, *false negative* rate, and anomaly detection rate for varying rates of malicious traffic. From the results in Fig. 11(a), we can see that for higher rates of malicious traffic, both the overhead and the detection time increase. Fig. 11(b) illustrates that with a capacity of $1Gb/s$ for the sampling switches Cyber-AnDe improves detection performance with respect to *false negative* by around 90% in comparison to fixed sampling.

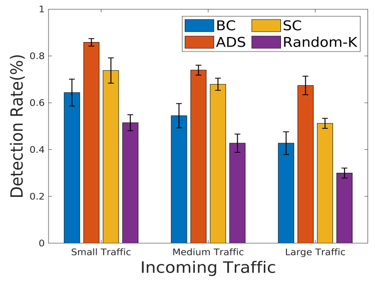
Fig. 11(c) presents results comparing the performance of anomaly detection rate of our proposed Cyber-AnDe and Fixed sampling. Since some normal flows can exhibit behavior very similar to that of a malicious flow, it is impractical to develop a model with 100% detection accuracy. The proposed Cyber-AnDe has significantly better results in detecting malicious flows since it takes into consideration the capacity limitation of the switches. Note that the sampling capacity of each switch is limited. When the controller requests sampling from a switch at a rate that exceeds the switch's capacity, the switch will be unable to handle this request. This will result in a smaller sample size than what the controller expects. However, by reassigning the higher rate sampling tasks to other switches that have the capacity, our framework effectively achieves better performance.

In Fig. 12, we present the effect of traffic scale and capacity of sampling switches on the performance of CyberAnDe compared to other fixed sampling and OpenFlow methods. Fig. 12(a) shows the minimum number of sampling switches required to achieve a minimum required anomaly detection rate of 90% or higher for the three methods for varying flow rates. Also, an MTR of 2.5% is considered in this case. From the results, we can conclude that among the three methods, Cyber-AnDe has the best performance. Cyber-AnDe achieves the required 90% or higher anomaly detection with much fewer switches compared to the other two methods. For example, for 280 flow, Cyber-AnDe requires only 4 switches compared to OpenFlow and Fixed methods require 7 and 12 switches respectively. Even when the number of flows increases from 260 to 320, Cyber-AnDe can achieve the minimum required anomaly detection rate of 90% or higher without the need for any additional switches. In conclusion, the minimum number of required sampling switches for an anomaly detection rate of 90% or higher is on average 48% lower with Cyber-AnDe.

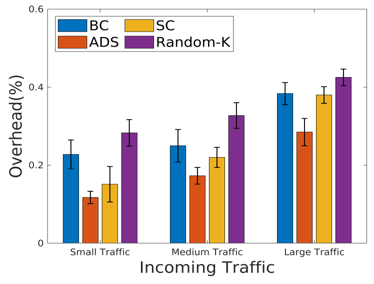
Fig. 12(b) illustrates the overhead of controllers under different volumes of incoming traffic with a fixed MTR of 2.5%. While the control messages generated by our framework increase with increasing volume of incoming traffic, it still is significantly lower compared to OpenFlow and Fixed sampling techniques. Once again, from the results it can be seen that of the three methods Cyber-AnDe has the best performance. The controller overhead with Cyber-AnDe is 42% lower compared to OpenFlow and 26% lower compared to Fixed sampling. Fig. 12(c) shows the impact of the capacity of sampling switches on the *false negative* rate with a fixed MTR of 2.5%. With our framework, the *false negative* rate is around 78%



(a) Detection Time - MTR: 2.5%

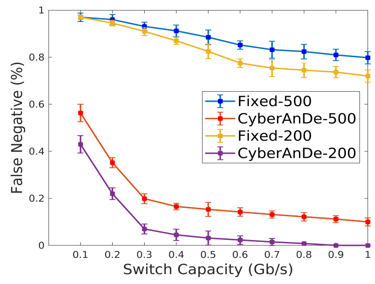


(b) Detection Rate - MTR: 2.5%

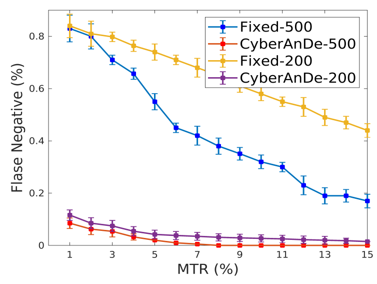


(c) Overhead - MTR: 2.5%

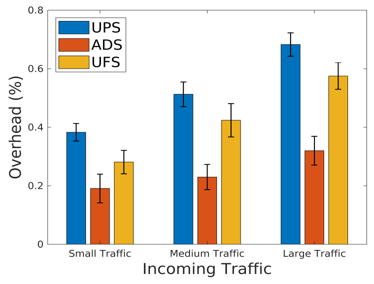
Fig. 13: Comparing performance of different sampling switch selecting methods when $\gamma_0 = 1/100$.



(a) False Negative for different switch capacities



(b) False Negative for different MTR



(c) Overhead for varying traffic volumes

Fig. 14: Evaluating method on different switch capacities, MTR, and controller overhead when $\gamma_0 = 1/100$.

lower for switches with $0.2Gb/s$ capacity, around 95% lower for switches with $0.5Gb/s$ capacity, and between 99%-100% lower for switches with capacity $\geq 0.9Gb/s$.

In Fig. 13, we compare the detection time performance of Cyber-AnDe's ADS with three other popular switch selection algorithms – BC, SC, and Random-k. Additionally, we consider three scenarios under a fixed incoming malicious traffic rate (MTR) of 2.5%. From the results in Fig. 13(a), it can be seen that our sampling technique achieves the fastest detection time. Fig. 13(b) presents the detection rate performance of four methods we have implemented for selecting sampling switches. From the results, it can be seen that our framework's sampling switch selection algorithm has the best performance, i.e., the most accurate anomaly detection.

Fig. 13(c) presents the overhead performance of four methods we have implemented for selecting sampling switches. It can be seen from the results that Cyber-AnDe's sampling switch selection algorithm adds the lowest overhead on the controller while achieving the highest anomaly detection rate. We consider a fixed MTR to evaluate the four methods for selecting switches for traffic sampling.

Fig. 14(a) compares the detection rate performance of Cyber-AnDe' ADS to fixed sampling method. From the results shown, it can be concluded that our ADS has better anomaly detection performance because it considers the capacity limitation of the switches when selecting the sampling switches. Fig. 14(b) shows the impact of the capacity of sampling switches on the FN. We have considered a fixed malicious rate of 2.5% for this simulation. It can be seen that Cyber-AnDe has significantly better performance in both cases, i.e., 200 and 500 flows.

Fig. 14(c) shows the amount of controller's overhead for different sampling methods: 1)Uniform Packet Sampling (UPS): Each switch samples every arriving packet with probability p. UPS is a representative of per-port sampling solutions and resembles sFlow [34], 2)Uniform Flow Sampling (UFS): Each switch samples every flow with rate p. Once a switch has captured p fraction of a flow, it stops sampling that flow. To achieve this, UPS assumes a priori knowledge about flow rates. UFS represents uniform per-flow sampling solutions and resembles NetFlow [35], and 3)Adaptive Distributed Sampling (ADS): The proposed sampling method. In this experiment, we assumed $p = 0.1$. It can be seen that Cyber-AnDe's ADS has the best performance as it has the least amount of overhead.

In this section, we present the results of evaluation on a DDoS dataset, HLD-DDoSDN, in [36] which includes three types of DDoS attacks: 1) High and low rates TCP DDoS attacks, 2) High and low-rate UDP DDoS, and 3) High and low-rate ICMP DDoS attacks. According to [37], a sending packet rate of 0.2(s) and 0.03(s) indicate high-rate and low-rate DDoS flooding attacks, respectively.

Table V-B presents a comparison of recent sampling methods for SDN which are Spatial-temporal Collaborative Sampling (STCS) [30] and FlowShark [31] with Cyber-AnDe over the HLD-DDoSDN dataset. We consider traffic with different rates and protocols to demonstrate the performance of these methods in various scenarios. The results in Table V-B indicate that Cyber-AnDe consistently outperforms the other methods across all traffic conditions. Specifically, Cyber-AnDe achieves the lowest detection times, with values ranging from $21.32ms$ to $37.21ms$, indicating its superior speed in identifying DDoS

Comparing recent sampling methods over variation of DDoS attack on SDN.

Sampling Approaches	**Detection Time**(ms)				**Detection Rate**$(\%)$				**Overhead**$(\%)$			
	High Rate TCP	**Low Rate TCP**	**High Rate UDP**	**Low Rate UDP**	**High Rate TCP**	**Low Rate TCP**	**High Rate UDP**	**Low Rate UDP**	**High Rate TCP**	**Low Rate TCP**	**High Rate UDP**	**Low Rate UDP**
STCS [30]	28.31	34.28	37.12	43.5	93.23	89.88	88.5	87.78	0.17	0.28	0.25	0.32
FlowShark [31]	25.61	31.31	35.85	41.43	95.11	92.78	91.34	90.67	0.13	0.20	0.21	0.26
Cyber-AnDe	21.32	28.15	32.5	37.21	97.46	95.72	95.29	93.45	0.14	0.22	0.18	0.25

TABLE II: Evaluation of Cyber-AnDe over different network topologies when $\gamma_0 = 1/100$, $MTR = 2.5\%$.

Network Topology	**Detection rate**$(\%)$		**Overhead**$(\%)$	
	Small Traffic	**Large Traffic**	**Small Traffic**	**Large Traffic**
Star	0.83	0.65	0.25	0.43
Ring	0.83	0.65	0.28	0.46
Mesh	0.87	0.71	0.18	0.30
Fat-tree	0.90	0.87	0.18	0.31

attacks. Furthermore, Cyber-AnDe demonstrates the highest detection rates, maintaining above 93.45% across all scenarios, which underscores its reliability and accuracy in detecting threats. In terms of overhead, Cyber-AnDe maintains a low impact on system resources, with values comparable to those of FlowShark and slightly higher than STCS, demonstrating a balanced trade-off between performance and resource consumption. These results suggest that Cyber-AnDe is a robust and efficient solution for DDoS attack detection in SDN environments, offering a combination of fast response times and high detection accuracy with a manageable overhead.

Table II presents the evaluation of Cyber-AnDe over different network topologies, given $\gamma = 1/100$ and $MTR = 2.5\%$. Results demonstrate that the Fat-tree topology outperforms the others by offering the highest detection rates and maintaining low overheads, making it the most efficient choice. The Mesh topology also shows strong performance with good detection rates and low overheads. In comparison, the Star and Ring topologies have lower detection rates and slightly higher overheads, indicating that they are less effective in managing both small and large traffic scenarios. Overall, the Fat-tree topology provides the best balance of high detection rates and low overheads, making it the most efficient among the evaluated topologies.

VI. Discussion

Although the study aims to show that the algorithm can be used universally, there are still potential threats to its validity. These threats come from both practical application issues and limitations within the algorithm itself. These concerns include the compromise or failure of the SDN controller. This paper assumes that the SDN controller within the Cyber-AnDe framework is trustworthy and securely managed. While we acknowledge the risks associated with having a centralized controller, our focus is on exploring adaptive sampling strategies for anomaly detection within this secure framework. Another concern is the performance of the individual components within the Cyber-AnDe framework. To this end, the success of the proposed solution depends on how well both the RL agent and the BMA module perform. Additionally, the ability of attackers presents a threat. An adaptive attacker who knows the mechanism may manipulate network traffic to evade detection by the IDS.

This paper discusses routine attacks without directly addressing adaptive adversaries, but acknowledges their importance in real-world situations. It proposes a system that can adapt, thanks to a controller and a RL agent, allowing updates to sampling scenarios in response to changing threats. Future research will explore how well the proposed system works against adaptive adversaries, including multi-stage attackers to thoroughly assess its robustness under different threat conditions.

This paper primarily focused on assessing the system's runtime overhead but acknowledges the importance of considering memory limitations in real-world deployment scenarios. It proposes a strategy to allocate sampling switch assistance to devices with potential memory constraints to mitigate overload and ensure feasibility. While memory overhead is considered, the paper's main objectives are to evaluate Cyber-AnDe's detection rate and time, crucial for its effectiveness as an intrusion detection system. Future research will explore how well the proposed system works against adaptive adversaries, including multi-stage attackers to thoroughly assess its robustness under different threat conditions.

VII. Conclusion

Effective network monitoring plays an increasingly critical role in the overall security posture of an enterprise. SDNs are very promising and have the potential to enable robust security solutions if their ability to decouple the control plane and data plane can be effectively leveraged. Leveraging the SDNs' programmable functionality of OpenFlow switches, incoming traffic can be selectively sampled by the controller. However, it is vital that sampling on SDN switches be carefully managed such that the anomaly detection rate is maximized while keeping the overhead and congestion to a minimum. In this paper, we propose Cyber-AnDe – a robust cybersecurity framework with an adaptive distributed sampling method for network traffic-based anomaly detection on SDNs. Through simulations and SDN test-bed-based experiments, we evaluate the performance of Cyber-AnDe in terms of *detection time*, *detection rate*, and *overhead*.

Results confirm that our solution significantly outperforms existing methods, demonstrating a 93% improvement in network traffic-based anomaly detection. Cyber-AnDe achieves the required detection accuracy of 90% or higher while utilizing 48% fewer switches on average. It also reduces overhead by an average of 34% and improves false-negative rates by at least 78%. In conclusion, our proposed Cyber-AnDe framework offers superior anomaly detection performance compared to current solutions.

References

[1] J. H. Cox, J. Chung, S. Donovan, J. Ivey, R. J. Clark, G. Riley, and H. L. Owen, "Advancing software-defined networks: A survey," *IEEE Access*, vol. 5, pp. 25 487–25 526, 2017.

[2] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," *Computer Networks*, vol. 75, pp. 453–471, 2014.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[4] Y. Hande, A. Muddana, and S. Darade, "Software-defined network-based intrusion detection system," in *Innovations in Electronics and Communication Engineering*. Springer, 2018, pp. 535–543.

[5] Niknami, Nadia and Wu, Jie, "Enhancing load balancing by intrusion detection system chain on sdn data plane," in *Proc. of IEEE Conf. on Communications and Network Security (CNS)*, 2022, pp. 264–272.

[6] S. Kaur, K. Kumar, and N. Aggarwal, "Analysis of ddos attacks in software defined networking," in *IEEE Delhi Section Conf. (DELCON)*, 2022, pp. 1–6.

[7] S. Yang, L. Cui, Z. Chen, and W. Xiao, "An efficient approach to robust sdn controller placement for security," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1669–1682, 2020.

[8] Z. Wang, Y. Li, and S. Guan, "A robust-link controller placement model for large-scale software defined networks," *Transactions on Emerging Telecommunications Technologies*, vol. 34, no. 6, p. e4765, 2023.

[9] K. Bartos and M. Rehak, "Towards efficient flow sampling technique for anomaly detection," in *International Workshop on Traffic Monitoring and Analysis*. Springer, 2012, pp. 93–106.

[10] R. M. A. Ujjan, Z. Pervez, K. Dahal, A. K. Bashir, R. Mumtaz, and J. González, "Towards sflow and adaptive polling sampling for deep learning based ddos detection in sdn," *Future Generation Computer Systems*, vol. 111, pp. 763–779, 2020.

[11] S. Yoon, T. Ha, S. Kim, and H. Lim, "Scalable traffic sampling using centrality measure on software-defined networks," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 43–49, 2017.

[12] T. M. Chan, E. Grant, J. Könemann, and M. Sharpe, "Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling," in *Proc. of the 23rd annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 2012, pp. 1576–1585.

[13] N. Bansal, R. Krishnaswamy, and B. Saha, "On capacitated set cover problems," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2011, pp. 38–49.

[14] A. Gupta, R. Krishnaswamy, A. Kumar, and D. Panigrahi, "Online and dynamic algorithms for set cover," in *Proc. of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 2017, pp. 537–550.

[15] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[16] W. Zulehner, "Analysis of iterative methods for saddle point problems: a unified approach," *Math. Comput.*, vol. 71, pp. 479–505, 2002.

[17] A. Patcha and J.-M. Park, "An adaptive sampling algorithm with applications to denial-of-service attack detection," in *Proc. of 15th IEEE Intl. Conf. on Computer Communications and Networks*, 2006, pp. 11–16.

[18] Z. Ning and J. Gong, "A sampling method for intrusion detection system," in *Asia-Pacific Network Operations and Management Symposium*. Springer, 2008, pp. 419–428.

[19] S. Ali, I. U. Haq, S. Rizvi, N. Rasheed, U. Sarfraz, S. A. Khayam, and F. Mirza, "On mitigating sampling-induced accuracy loss in traffic anomaly detection systems," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 3, pp. 4–16, 2010.

[20] R. Cohen and E. Moroshko, "Sampling-on-demand in sdn," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2612–2622, 2018.

[21] A. Campazas-Vega, I. S. Crespo-Martínez, Á. M. Guerrero-Higueras, C. Álvarez-Aparicio, V. Matellán, and C. Fernández-Llamas, "Analyzing the influence of the sampling rate in the detection of malicious traffic on flow data," *Computer Networks*, vol. 235, p. 109951, 2023.

[22] Y. Du, H. Huang, Y.-E. Sun, S. Chen, and G. Gao, "Self-adaptive sampling for network traffic measurement," in *Proc. of the IEEE Conf. on Computer Communications*, 2021, pp. 1–10.

[23] R. Biswas, S. Kim, and J. Wu, "Sampling rate distribution for flow monitoring and ddos detection in datacenter," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2524–2534, 2021.

[24] P. Manso, J. Moura, and C. Serrão, "Sdn-based intrusion detection system for early detection and mitigation of ddos attacks," *Information*, vol. 10, no. 3, p. 106, 2019.

[25] A. Yazdinejadna, R. M. Parizi, A. Dehghantanha, and M. S. Khan, "A kangaroo-based intrusion detection system on software-defined networks," *Computer Networks*, vol. 184, p. 107688, 2021.

[26] N. Niknami and J. Wu, "Entropy-kl-ml: Enhancing the entropy-kl-based anomaly detection on software-defined networks," *IEEE Transactions on Network Science and Engineering*, 2022.

[27] Niknami, Nadia and Wu, Jie, "Idsmatch: A novel deployment method for ids chains in sdns," *Network*, vol. 4, no. 1, pp. 48–67, 2024.

[28] D. Yeom, N. Niknami, and J. Wu, "Evaluating performance of intrusion detection systems under different configurations in sdn," ser. MobiHoc '23. Association for Computing Machinery, 2023, p. 418–423.

[29] S. Kim, S. Yoon, and H. Lim, "Deep reinforcement learning-based traffic sampling for multiple traffic analyzers on software-defined networks," *IEEE Access*, vol. 9, pp. 47 815–47 827, 2021.

[30] 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 Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 999–1013, 2020.

[31] S. Sadrhaghighi, M. Dolati, M. Ghaderi, and A. Khonsari, "Flowshark: Sampling for high flow visibility in sdns," in *Proc. of IEEE Conf. on Computer Communications (INFOCOM)*. IEEE, 2022, pp. 160–169.

[32] H. Cai, J. Deng, S. Chen, X. Wang, S. Pack, and Z. Han, "Improved flow awareness by spatio-temporal collaborative sampling in software defined networks," in *Proceeding of the IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[33] L. Lü, D. Chen, X.-L. Ren, Q.-M. Zhang, Y.-C. Zhang, and T. Zhou, "Vital nodes identification in complex networks," *Physics Reports*, vol. 650, pp. 1–63, 2016.

[34] sFlow, "Making the network visible," https://sflow.org/, accessed: 2022-01-10.

[35] B. Claise, "Cisco systems netflow services export v9," https://www.ietf.org/rfc/rfc3954.txt, accessed: 2022-01-10.

[36] A. A. Bahashwan, M. Anbar, S. Manickam, G. Issa, M. A. Aladaileh, B. A. Alabsi, and S. D. A. Rihan, "Hld-ddosdn: High and low-rates dataset-based ddos attacks against sdn," *Plos one*, vol. 19, no. 2, p. e0297548, 2024.

[37] M. A. Aladaileh, M. Anbar, A. J. Hintaw, I. H. Hasbullah, A. A. Bahashwan, and S. Al-Sarawi, "Renyi joint entropy-based dynamic threshold approach to detect ddos attacks against sdn controller with various traffic rates," *Applied Sciences*, vol. 12, no. 12, p. 6127, 2022.