SmartTC: A Real-Time ML-Based Traffic Classification with SmartNIC

Lingxiang Hu[△†], Chenyang Hei[△†], Fuliang Li[†✉], Chengxi Gao[£✉], Jiaxing Shen[‡], Xingwei Wang[†]
[†]Northeastern University, China
[£]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, [‡]Lingnan University
Email: hulingxiang@stumail.neu.edu.cn, 2310726@stu.neu.edu.cn, lifuliang@cse.neu.edu.cn,
chengxi.gao@siat.ac.cn, jiaxingshen@LN.edu.hk, wangxw@mail.neu.edu.cn

***Abstract*—Real-time network traffic classification plays a crucial role in ensuring Quality of Service and network security, and machine learning (ML) based methods achieve high classification accuracy but induce significant computational overhead. While SmartNIC solutions can offload classification tasks thus reducing CPU burdens, they still suffer from various limitations, manifested in limited computing capabilities, insufficiency in dynamic load handling and high latency from heterogeneous computing architectures.**

To solve these problems, we propose *SmartTC*, with three key designs: (1) *SmartTC* employs hardware-software co-design to optimize SmartNIC processing power, (2) *SmartTC* adopts a traffic-aware dynamic batch submission strategy that adjusts submission policies based on real-time network load, and (3) *SmartTC* proposes parallel pipeline scheduling that ensures efficient task execution while minimizing communication overhead. Finally, we implement *SmartTC* on the BlueField-3 DPU and conduct extensive experiments for evaluations, and comparison results demonstrate that *SmartTC* significantly outperforms existing solutions. For example, it reduces average traffic classification time by up to 16.8% under low loads and 90.9% under high loads. Besides, *SmartTC* does not affect Bluefield-3 network services, and saves host CPU usage by at least two cores.

***Index Terms*—Traffic Classification, Machine Learning, SmartNIC**

I. Introduction

In the cloud environment nowadays, real-time traffic classification plays a crucial role in ensuring Quality of Service (QoS) [17] and identifying and isolating malicious traffic to enhance network security [3]. Currently, the primary approach to improving traffic classification accuracy is to employ ML-based classification models. However, as model complexity increases and data center scales expand, the computational resource demands for traffic classification tasks are growing exponentially, posing a critical bottleneck to the real-time performance of traffic classification systems. Therefore, it is essential to develop an efficient ML-based traffic classification system.

To address the aforementioned issue, existing solutions can be categorized into two types based on the deployment location of the traffic classification model. The first approach deploys the classification task on the network data plane [24] such as programmable switche or FPGA-based SmartNIC, including systems like Leo [8], NetBeacon [26], and N3IC [19]. These methods perform real-time traffic classification in the data plane, avoiding the high overhead associated with data transmission [22], thereby reducing latency and alleviating the computational burden on the host's CPU and GPU. However, due to the limited computational resources of programmable network devices, this approach can only deploy lightweight models and incurs high costs when updating ML models. The second approach executes traffic classification on the host CPU and GPU, as seen in systems like FENXI [5] [4], which typically follow a modular architecture comprising analysis, pre-processing, and traffic classification. This modular design allows for performance improvements by scaling host-side resources. However, such methods often require significant host resources, potentially impacting regular operations, and lack optimizations tailored to real-world deployment scenarios.

System-on-Chip (SoC) SmartNIC offers a novel solution for deploying ML-based traffic classification models, serving as a compromise among existing solutions. Compared to programmable switches and other programmable network devices, SoC SmartNICs provide greater computational power, alleviating the resource constraints of existing programmable network devices. Additionally, as an extension of host-side computing resources, SmartNIC can offload computational tasks, effectively reducing the CPU burden and improving traffic classification efficiency.

However, directly migrating and deploying traffic classification tasks onto SoC SmartNICs presents several challenges in real-world cloud environments, and the key limitations are manifested in the following areas:

Limitation 1: Difficulty in offloading entire traffic classification system to SmartNICs. Existing host-based traffic classification systems (e.g., FENXI [5]) usually require binding several CPU cores and inference devices such as GPUs. Although SoC SmartNICs have stronger computing capabilities than programmable network devices, their performance is still far inferior to that of the host's CPU [23] and GPU. Offloading the entire ML-based classification system to a SmartNIC may not only reduce the classification efficiency but also introduce additional processing overhead (refer to Challenge 1 in Section II.B for more details).

Limitation 2: Lack of mechanisms to handle dynamic network loads. Existing traffic classifiers such as FENXI optimize for static workloads through fixed-batch processing. In

[△]Both authors contributed equally to this paper.

this approach, input data is aggregated into pre-set batch sizes (e.g., 2 to 128 tasks) to reduce the number of kernel launches, maximize the utilization of the GPU, and thus improve the overall computational efficiency. However, network traffic in cloud environments fluctuates greatly: microsecond-level burst traffic coexists with sustained high-throughput traffic. This presents a fundamental trade-off problem for the fixed-batch processing strategy. On one hand, large batch processing can increase GPU utilization, but it will introduce waiting latency during periods of low network load. On the other hand, although small batch processing can reduce waiting latency, it will lead to frequent inferences during high network load periods, causing the GPU to start and switch tasks frequently. The lack of a mechanism to dynamically adjust the batch size to adapt to dynamic network loads reduces the real-time performance of traffic classification. A system that can sense network loads and perform scheduling is crucial for maintaining real-time performance under dynamic network loads (refer to Challenge 2 in Section II.B for more details).

Limitation 3: Additional latency from heterogeneous computing architectures. SmartNICs can offload computational tasks, reducing the computational pressure on the host's CPU and GPU and freeing up resources. However, in a heterogeneous computing architecture with the host's CPUs and GPUs that supports task offloading, transferring memory data from SmartNICs to the host requires transmission through the PCIe. This approach also introduces additional communication overhead (refer to Challenge 3 in Section II.B for more details). In addition, to ensure the collaboration between the SmartNIC and the host, it is necessary to synchronize the traffic classification model categories. Therefore, designing an efficient pipeline and communication mechanism is crucial to reduce latency and ensure synchronization of the model categories.

To address the aforementioned limitations, this study proposes ***SmartTC***, a real-time ML-based traffic classification system leveraging SmartNIC offloading. *SmartTC* introduces three key optimization strategies: **(1) Hardware-software co-design**: To address the limitation of SmartNIC computing resources, *SmartTC* adopts a partial offloading approach, where the analysis and pre-processing modules of the traffic classification system are offloaded to the SmartNIC. Meanwhile, hardware accelerators on the SmartNIC enable efficient network traffic processing, while software components execute traffic analysis in parallel. By implementing hardware-software co-design on the SmartNIC, the system ensures efficient execution of analysis and pre-processing. **(2) Traffic-aware dynamic batch submission strategy**: To handle dynamic changes in network load, *SmartTC* introduces a traffic-aware dynamic batch submission strategy. This strategy utilizes long-term and short-term windows to monitor both instantaneous and overall network traffic, dynamically adjusting batch submission policies to accommodate different load conditions, thereby improving real-time performance. **(3) Parallel pipeline scheduling**: To mitigate additional communication latency introduced by the heterogeneous computing architecture, *SmartTC* employs a parallel pipeline scheduling method. This approach organizes the execution of tasks into a pipeline between the SmartNIC and the host CPU and GPU, ensuring efficient parallel execution across these components. By overlapping computation with communication, this approach effectively minimizes overall system latency and improves traffic classification efficiency.

The main contributions of this study are as follows:

- We first analyze the limitations of existing work and evaluate the applicability of SmartNIC-based offloading for traffic classification.
- We develop *SmartTC* on the BlueField-3 DPU. First, hardware-software co-design is adopted to improve the efficiency of analysis and pre-processing. Second, a traffic-aware dynamic batch submission strategy is introduced to enhance the system's adaptability to dynamic network loads. Finally, a parallel pipeline scheduling method is designed to optimize system throughput and improve traffic classification efficiency.
- Finally, we conduct extensive experiments to evaluate the performance of *SmartTC*, and comparison results show that *SmartTC* significantly improves traffic classification efficiency and effectively reduces host resource consumption. For example, *SmartTC* outperforms the fixed batch strategy under different network loads, reducing the average flow classification time by up to 16.8% under low loads and 90.9% under high loads. In the meanwhile, *SmartTC* ensures that network services remain unaffected and reduces CPU usage by at least two cores.

II. Background and Motivation

A. Background

ML for traffic classification: Typically, network traffic characteristics are matched against a known feature library of applications to identify the various applications running on a network. Traditional port-based classification methods rely on predefined mappings between port numbers and application types [16], such as port 80 typically corresponds to web browsing applications, or analysis based on network payloads. However, with the increasing prevalence of encrypted traffic, these methods are gradually becoming ineffective. ML-based approaches can identify and classify traffic from more dimensions, as discussed in [11], [13], [12], and [1]. These models can automatically learn effective features from complex network data, thereby improving classification accuracy and robustness. For example, the DF model [20] utilizes convolutional neural networks (CNNs) to process raw packet information, enabling efficient classification of encrypted traffic. Additionally, the SAM [25] model uses a self-attention mechanism to model traffic, further enhancing classification performance. Although the increasing complexity of the models has led to significant improvements in the accuracy of ML-based traffic classification, this complexity has also resulted in the neglect of real-time requirements. Therefore, an efficient traffic classification system must be designed in practical deployments to address the challenges of real-time performance.

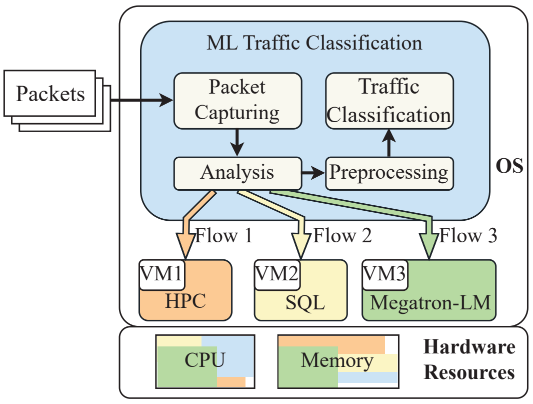

Fig. 1. In a multi-tenant cloud environment, the traffic classification system is responsible for classifying network traffic. The system is deployed on the same host as other tasks, sharing the host's resources.

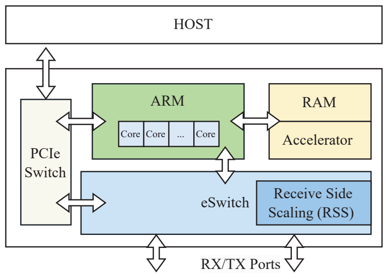

Fig. 2. Bluefield-3 DPU architectures.The figure illustrates that the DPU is equipped with multiple ARM cores and associated acceleration hardware.

Systems for real-time ML-based traffic classification: With the increase in model complexity and the expansion of data center scale, the computational requirements have significantly increased. Offline traffic classification methods lead to high response latency. For classification tasks that require quick responses, this latency is unacceptable. Therefore, one current approach is to perform traffic analysis tasks on the network data plane. N3IC [19], Leo [8], and Switchtree [10] embed ML models into the data plane. By executing on the data plane, low latency is achieved, and host resources are freed up. However, the SRAM of programmable network devices used in the above studies typically only has tens of MBs [15] [14], which limits the deployment of models with parameters at the k-level [8] [19], preventing the deployment of models with parameters at the m-level (as shown in Table II). Another common solution is to deploy traffic classification on host CPUs and GPUs. To effectively deploy these ML models, [4] and [5] typically adopt a modular architecture, as shown in Figure 1. The modular design makes it easy to expand and can meet the resource requirements of different ML-based traffic classification tasks. This system ensures the real-time performance of traffic classification but causes significant consumption of host resources. This situation creates a demand for a system that can not only guarantee real-time performance but also minimize resource utilization.

SmartNIC: The emergence of SmartNIC[*] has provided additional computing resources and task offloading platforms. SoC SmartNIC architectures have been widely adopted, with the BlueField series of chips [2] serving as a prominent example. As shown in Figure 2, their architecture integrates ARM processor cores. The eSwitch is equipped with a variety of hardware accelerators and can also forward traffic to the ARM subsystem for more flexible software processing. As a result, this processing pipeline offers significant flexibility for network packet handling. Existing research leverages SoC SmartNIC for offloading and acceleration. For instance, SKV [21] offloads data copy operations to SmartNIC to optimize distributed key-value storage, thereby improving backend processing efficiency, increasing throughput, and reducing latency. LineFS [9] introduces a SmartNIC-based distributed file system that reduces CPU load by offloading computational tasks. SmartEmb [18] utilizes SmartNIC to accelerate inference, employing techniques such as task reordering, prefetching, and cache management to optimize embedding table lookups, thereby reducing CPU contention. Additionally, Magician [6] mitigates intra-host network congestion through hot data offloading, dynamic update strategies, and server-side consistency mechanisms, thereby enhancing the performance of network applications. SmartNIC has achieved remarkable results in offloading host resources. Integrating SmartNIC with the traffic classification system can effectively relieve host resource pressure. However, to meet real-time requirements, in scenarios with constantly changing network loads, how to make full use of SmartNIC resources and explore more efficient collaborative cooperation methods between SmartNIC and host has brought a series of challenges that urgently need to be overcome.

[*]In this paper, the terms DPU (Data Processing Unit) and SmartNIC will be used interchangeably, both referring to SoC SmartNIC.

B. *Motivation*

While SmartNIC can offload computing tasks from CPUs, its application to traffic classification still faces various challenges as follows.

Challenge 1: Resource constraints. Although SoC SmartNICs can run more complex models than programmable network devices (with k-level parameters), their inference speed remains significantly slower than that of GPUs—up to 10 times slower, as shown in Table I and Table II. Therefore, offloading the entire ML-based classification system to a SmartNIC still remains a suboptimal solution.

Although the cost of a SmartNIC (e.g., Nvidia BlueField-3) may exceed that of several CPU cores, its capacity to integrate multiple offload tasks—including storage, security, and networking—has driven its widespread adoption in modern cloud infrastructures. The associated cost can be effectively amortized across different tasks.

TABLE I
COMPARISON OF DIFFERENT MODELS (TSRNN, SAM, MATEC)

Parameter	TSRNN	SAM	MATEC
Parameters	2.9M	0.8M	2.6M
Input Size	22.5KB	0.1KB	23.58KB
Inference Time (GPU)	2.98ms	0.59ms	1.34ms
Inference Time (DPU)	26.70ms	4.91ms	10.56ms

TABLE II
COMPARISON OF DIFFERENT MODELS (ET-BERT, LEO, N3IC)

Parameter	ET-BERT	Leo	N3IC
Parameters	132M	1K	8.8k-41.5k
Input Size	393KB	X	X
Inference Time (GPU)	13.47ms	X	X
Inference Time (DPU)	109.54ms	X	X

Challenge 2: Dynamic network loads. Existing methods lack effective mechanisms to adapt to dynamically changing network loads. FENXI [5] relies on fixed batch sizes for inference. While this approach can improve overall hardware utilization and increase inference throughput, it presents a dual challenge of balancing real-time performance and resource utilization under dynamic network loads. As illustrated in Figure 3, using small batch processing enables real-time inference for each batch; however, under high network load, it triggers frequent inference requests, causing continuous context switching on the GPU and increasing computational overhead. Conversely, using large batch processing can fully utilize computational resources but leads to an increase in average classification time under low network load due to network packet waiting, thereby compromising the real-time performance of traffic classification. In addition, the FENXI approach requires allocating at least 2–4 CPU cores to maintain network performance, resulting in high resource consumption.

Challenge 3: High latency in heterogeneous architectures. Although leveraging SmartNICs for computation offloading can effectively free up host resources, it faces critical challenges when building heterogeneous computing systems. Experiments shown in Figure 4 reveal that when the data transmission between the SmartNIC and the host reaches 256 KB, the combined communication latency caused by PCIe bus transmission and protocol stack processing rises to the millisecond level. This increasing communication overhead with data scale growth fails to meet the real-time requirements of traffic classification, resulting in overall system efficiency degradation.

III. DESIGN

In this section, we introduce our proposed traffic classification system, *SmartTC*, which combines SmartNICs with host. *SmartTC* employs hardware-software co-design and proposes two strategies: traffic-aware dynamic batch submission strategy and parallel pipeline scheduling.

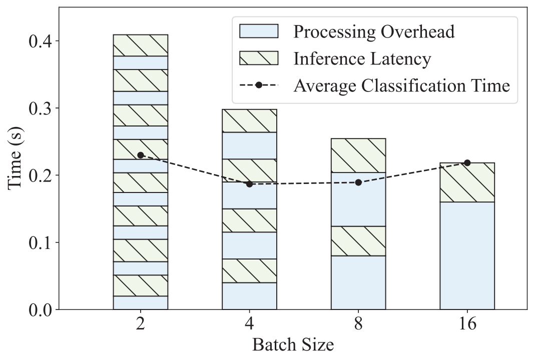

Fig. 3. As can be seen from the diagram, the batch processing method can indeed enhance the inference performance. However, in the scenario of network traffic classification, due to the extremely high requirements for real-time response in this scenario, increasing the batch size cannot continuously reduce the average classification time. In addition, the network load is in a state of dynamic change, which makes it an extremely challenging task to quickly determine the optimal batch size under dynamic network loads.

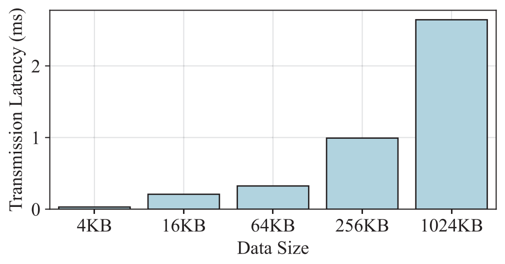

Fig. 4. As illustrated in the diagram, the latency of transferring data of varying sizes from the SmartNIC to the host increases as the data size grows.

A. Overview

As shown in Figure 5, the architecture of *SmartTC* primarily consists of an analysis module and a pre-processing module running on the SmartNIC, as well as a traffic classification module running on the host's CPU and GPU. When packets arrive at the SmartNIC, they are processed sequentially through the aforementioned modules. The functions of each module are summarized as follows:

- **Analysis Module** is responsible for polling and processing network traffic. The SmartNIC forwards packets to the analysis module running on the ARM subsystem to process the incoming packets
- **Pre-processing Module** handles the collected raw packets and submits the results to the subsequent module. The system processes the raw packets according to the model requirements and passes the processed data to the traffic classification module.
- **Traffic Classification Module** performs the model inference task, receiving the data from the pre-processing module and conducting classification on the host GPU.

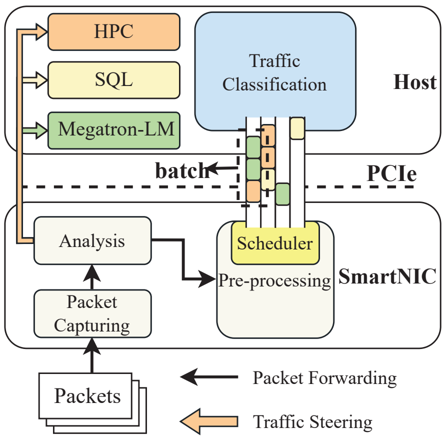

Fig. 5. This diagram illustrates the *SmartTC* architecture, which includes analysis and pre-processing modules executed on the SmartNIC, as well as a traffic classification module executed on the host.

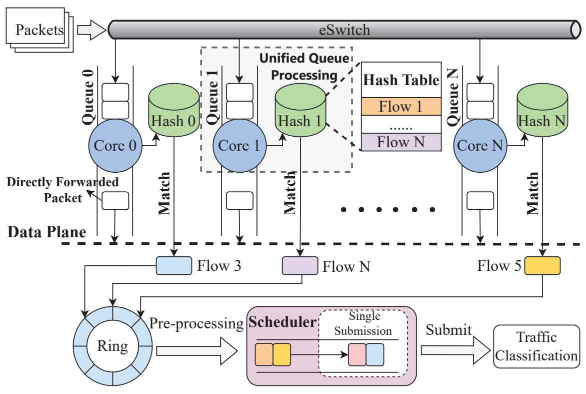

Fig. 6. The diagram illustrates the workflow of the analysis and pre-processing modules executed by the SmartNIC.

B. Analysis Module

The analysis module is designed to continuously monitor network traffic in real time through a polling mechanism and collect packets as required by the classification model. To achieve this, *SmartTC* offloads the module onto the SmartNIC. Its core mechanisms include:

- Flow tracking: Maintains the state of network connections, determining whether a packet belongs to an existing flow or a new flow.
- Packet forwarding and storage: Decides whether to store or forward a packet based on the requirements of the classification model.

As shown in Figure 6, when a network packet arrives at the module, it uses a hash bucket to track the state of the flow to which the packet belongs and identifies whether it is part of an existing flow or a new one. If it is a new flow, a new record is created in the hash bucket to store the flow state, and the packet is marked as the first packet of that flow. If it is part of an existing flow, the corresponding state in the hash bucket is updated, marking the current packet as the n-th packet of the flow, and its processing is determined based on the model requirements. Specifically, if the model requires the first θ packets of the flow and the current packet is within this range, it is first stored and then forwarded. Otherwise, if the packet arrives after the θ-th packet, it is forwarded directly without being stored.

We implement this mechanism on the SmartNIC through the following hardware-software co-design strategies:

- Hardware-accelerated flow distribution: Leveraging the *Receive Side Scaling (RSS)* [7] hardware feature of the SmartNIC, hash calculations are performed to automatically identify the flow to which each packet belongs and distribute them across different processing queues.
- Multi-core parallel architecture: Utilizing the SmartNIC's integrated multi-core ARM processors, a multi-queue mechanism is employed to achieve parallel processing of network traffic.

Although packet processing has significantly lower computational intensity compared to model inference, its continuous polling nature inevitably consumes a substantial amount of the host CPU's computational resources. To address this, SmartTC offloads such low-complexity but continuously running tasks to the SmartNIC, effectively reducing the host's resource burden and enhancing overall system efficiency.

C. Pre-processing Module

The pre-processing module is designed to efficiently process collected network packets and submit standardized inputs to the traffic classification module. *SmartTC* offloads the pre-processing module to the SmartNIC to reduce computation overhead on the host. This module consists of two core functions:

- Packet processing component: Standardizes collected network packets to meet the input requirements of traffic classification models.
- Traffic-aware dynamic batch submission strategy: Adjusts batch sizes dynamically based on real-time network traffic to adapt to varying network loads.

Packet Processing Component: After the analysis module accumulates a sufficient number of packets, the pre-processing module converts them into a format recognizable by the model. *SmartTC* summarizes the common packet processing requirements of existing traffic classification models into the following three standardization steps:

- Truncation: Removes redundant header information to ensure input format consistency.
- Padding: Adjusts packet lengths to a fixed size required by the model input.
- Normalization: Scales values to optimize model stability and inference performance.

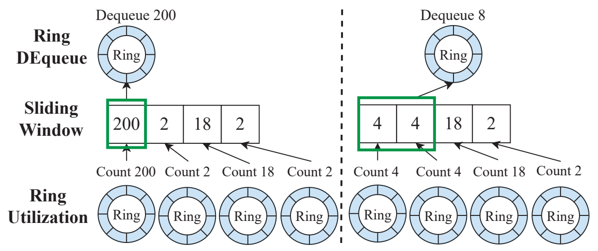

Fig. 7. The diagram illustrates the submission methods of the traffic-aware dynamic batch submission strategy across different scenarios.

Traffic-Aware Dynamic Batch Submission Strategy: After packet processing is completed, the system needs to submit batches of data conforming to the model's input format. However, fixed batch processing strategies struggle to adapt to dynamic network load. To address this issue, *SmartTC* proposes a traffic-aware dynamic batch submission strategy that dynamically adjusts batch submission size by sensing real-time network load, thereby meeting the real-time requirements of traffic classification.

This strategy introduces a **Short-Term Window** and a **Long-Term Window** mechanism:

- The short-term window monitors real-time network load and determines whether to submit the batch immediately. If the load is high, submission is triggered immediately to minimize latency.
- The long-term window analyzes network traffic over an extended period to prevent excessive delays under low load conditions. If the short-term window repeatedly fails to trigger a submission, the long-term window forces the batch submission.

As illustrated in Figure 7, this strategy covers two typical scenarios:

1) High network load: If the short-term window detects high network load, submission is triggered immediately.
2) Sustained low load: If multiple consecutive short-term windows do not trigger submission, the long-term window enforces submission to prevent excessive delays.

Algorithm 1 details the workflow of the traffic-aware dynamic batch submission strategy. Lines 1-3 initialize the parameters, obtaining the short-term waiting time t_θ and the batch size b_θ determined by the model. In practice, b_θ is empirically initialized within the range of 8 to 32 based on deployment conditions, while t_θ is set to the estimated inference time of a single batch to ensure timely responsiveness. Lines 5-12 form a loop that continuously monitors the traffic and adjusts dynamically. A batch submission is triggered when the current batch size A_{new} exceeds the predefined batch threshold b_θ (meeting the short-term window condition) or when *window* equals 1 (meeting the long-term window condition), after which *window* is reset to 0. If no submission is triggered, the system waits for t_θ to receive more packets and sets the long-term window flag *window* to 1. This strategy adaptively

Algorithm 1 Traffic-Aware Dynamic Batch Submission

```
1:  Initialize $window \leftarrow 0$
2:  $t_\theta \leftarrow T(model)$
3:  $b_\theta \leftarrow B(model)$
4:  while True do
5:      $A_{new} \leftarrow \textsc{Count}(ring)$
6:      if $A_{new} > b_\theta$ or $window = 1$ then
7:          $window \leftarrow 0$
8:          $\textsc{Submit}(b_\theta)$
9:      else
10:         $\textsc{Wait}(t_\theta)$
11:         $window \leftarrow 1$
12:     end if
13: end while
```

adjusts the batch submission timing based on the current load, ensuring real-time response under low load and improved throughput under high load.

D. Traffic Classification Module

The traffic classification module is responsible for classifying preprocessed input data. In the *SmartTC* system architecture, the SmartNIC primarily handles analysis and preprocessing, while the traffic classification module, due to its higher computational requirements, is executed on the host using CPUs and GPUs. This heterogeneous computing architecture effectively offloads computation from the host. However, to ensure the efficient operation of the traffic classification module, *SmartTC* designs a SmartNIC-host communication protocol and a parallel pipeline scheduling mechanism to maximize system resource utilization and minimize computational and data transmission latency.

SmartNIC-Host Communication Protocol: To optimize the interaction between SmartNIC and the host, we design an efficient communication protocol that supports different traffic classification models deployment. Since different traffic classification models may have different input formats, this protocol introduces *Metadata* and *Batch Content* during data transmission to enhance flexibility and efficiency in data parsing and processing:

- Metadata: Contains batch size and traffic classification model information, ensuring that the host can correctly parse the model used for the current task and its corresponding batch data.
- Batch content: Stores actual inference input data, which is combined with metadata for parsing, supporting traffic classification tasks with different batch conditions.

This communication protocol enables SmartNIC to dynamically submit batch data while ensuring that the host can accurately parse and execute inference tasks. Under varying traffic classification tasks and batch conditions, the protocol provides a high level of generality.

Parallel Pipeline Scheduling: Heterogeneous computing architectures introduce additional communication latency. To

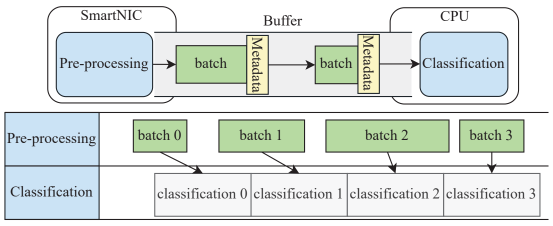

Fig. 8. The diagram shows how parallel pipeline scheduling hides the latency between the pre-processing module and the traffic classification module caused by the heterogeneous system.

reduce data transmission delays and improve system throughput, we propose a parallel pipeline scheduling that allows the pre-processing module and the traffic classification module to operate concurrently. This strategy incorporates buffer and pipeline execution to minimize synchronization blocking and optimize resource utilization.

As shown in Figure 8, the buffer stores batch data from the pre-processing module to prevent synchronization blocking between data transmission and inference tasks. Batch data transmitted by the SmartNIC is first buffered on the host, allowing the host to retrieve the next batch immediately after completing the current traffic classification task without waiting for new data transmission. This mechanism effectively reduces transmission bottlenecks and enhances the concurrency of classification tasks.

With the buffer in place, the traffic classification module on the host and the pre-processing module on the SmartNIC can execute concurrently, forming an efficient pipeline execution. The SmartNIC transmits new batch data to the host buffer while releasing completed batch tasks to free up storage space for new tasks. Simultaneously, the host-side traffic classification module directly extracts data from the buffer and executes traffic classification tasks on the GPUs. The SmartNIC continues to populate the buffer, ensuring seamless transitions between traffic classification tasks.

This parallel pipeline mechanism eliminates the sequential dependency between inference computation and data transmission, significantly reducing the communication latency introduced by the heterogeneous architecture.

IV. Evaluation

A. Evaluation Setup

Testbed: The experimental testbed consists of two high-performance servers including a worker node with *SmartTC* and a packet generation node, interconnected via an Ethernet switch. The worker node is equipped with an NVIDIA BlueField-3 [2], featuring a 16-core ARM processor, 32GB of onboard memory, and dual 200Gb/s network interfaces. It also includes an NVIDIA V100 GPU with 16GB of HBM memory. The packet generation node uses a Mellanox ConnectX-5 network card, providing network traffic through dual 100Gb/s interfaces. Both nodes are physically connected via a Mellanox SN2700 Ethernet switch, which supports 100Gb/s port speeds. All nodes run the Ubuntu 22.04 operating system, with DPDK version 22.03.0 deployed.

Implement: We implemented the *SmartTC* prototype system based on the Nvidia BlueField-3 SmartNIC. To maximize throughput and ensure real-time responsiveness, we allocated 10 ARM cores to run the analysis module, which processes incoming network traffic and stores packets addresses in a multi-producer ring queue. This approach enhances data throughput and reduces processing latency. Additionally, we allocated 1 ARM core to poll a single-consumer ring queue, which is dedicated to handling pre-processing tasks and dynamic batch submission, ensuring system stability and adaptive batch management. *SmartTC* executes the traffic classification module on the host. The module utilizes CPUs to receive batch inputs from the pre-processing module and subsequently perform traffic classification on the GPU. Additionally, *SmartTC* implements the smartNIC-host communication protocol at the application layer, based on Linux sockets. SmartTC uses a customized version of DPDK on the BlueField-3 subsystem to leverage the SmartNIC's multi-core ARM processors and hardware accelerators, avoiding the need to invoke host CPU resources as required by general-purpose NICs.

Metrics: Given that the impact of each component on traffic classification efficiency is difficult to isolate, we define the average time per flow after classification as the evaluation metric, i.e., the average classification time. The experiments are divided into two scenarios: low flow rate and high flow rate, to assess the performance optimization effect of *SmartTC*'s dynamic batch submission strategy.

Baselines: To the best of our knowledge, this is the first system that utilizes SoC SmartNIC for ML-based traffic classification. First, we select FENXI [5] as the most comparable system and use it as a baseline to evaluate our improvements in average classification time and resource consumption. Second, we compare *SmartTC* with BlueField-3 to assess its impact on network latency, demonstrating that *SmartTC* does not degrade existing network performance. SmartTC is a general-purpose traffic analysis system that supports the deployment of various traffic classification models. We deploy and evaluate three representative models: TSCRNN [11], MATEC [1], and SAM [25]. For the evaluation, we utilize Scapy to construct packets and tcpreplay to retransmit them under varying network loads. Although tcpreplay sends packets at a stable rate, the flows are generated randomly, which may cause short-term bursts in flow-level traffic. Network throughput is measured using iperf, latency is evaluated with netperf, and pktgen-dpdk is employed to assess packet forwarding capability.

B. Average Classification Time

This section analyzes the performance of the traffic-aware dynamic submission strategy under different network loads.

Figure 9 shows the performance comparison between *SmartTC* and FENXI in terms of batch submission. FENXI adopts a strategy of fixed batch sizes for submission, with the batch sizes X fixed at 2, 8, 32, and 128 respectively (named FENXI-X in Figure 9), which are used for comparison with

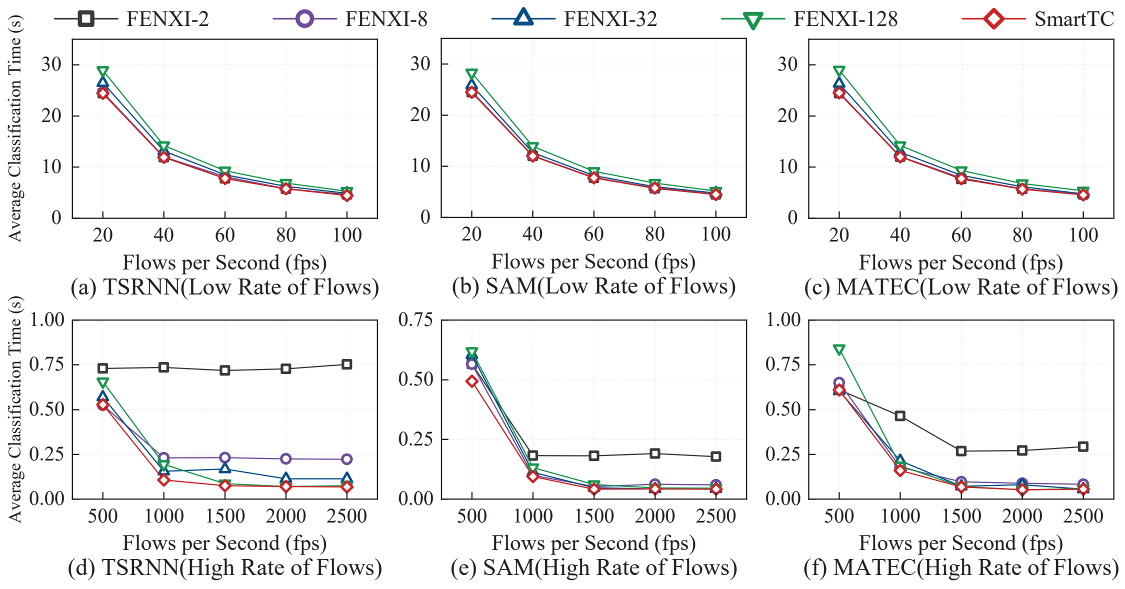

Fig. 9. The comparison of average classification time between the *SmartTC* system and FENXI under both low and high flow rate scenarios.

SmartTC. This experiment compares the average classification time to obtain the classification result, demonstrating the real-time requirements.

In low traffic scenarios (less than 100 flows per second), *SmartTC* and FENXI-2 outperform other batch strategies in terms of average classification time. Specifically, compared to other batch strategies using the TSCRNN model, *SmartTC* reduces the average classification time by up to 16.4%; for the MATEC model, the reduction is 16.8%; and for the SAM model, it is 13.7%. As the flow rate increases, the average classification time for FENXI-2 gradually converges to that of larger batch sizes. Across different models, *SmartTC* consistently achieves the shortest average classification time.

Under low network load, a small batch size can improve traffic classification efficiency. This is because before submission, a certain number of packets must be accumulated. Large batch strategies introduce processing delays, while small batch strategies submit data more promptly, reducing waiting time. *SmartTC* dynamically adjusts batch sizes based on network load, achieving more efficient task scheduling. As shown in Figure 9, this adaptation is particularly effective under low traffic rates.

In high traffic scenarios (more than 500 flows per second), *SmartTC* outperforms all fixed batch size strategies. Specifically, for the TSCRNN model, it reduces average classification time by up to 90.9%; for the MATEC model, by 80.9%; and for the SAM model, by 86.4%. As the flow rate continues to increase, *SmartTC*'s average classification time stabilizes.

However, the performance of FENXI-2 deteriorates in high-traffic scenarios due to increased communication overhead from frequent submissions. In such cases, larger batch sizes reduce the number of submissions, mitigating the impact of communication delays on average classification time. Nevertheless, as shown in Figure 9, merely increasing the batch size does not always yield lower average classification time. For example, at a flow rate of 500, the optimal batch sizes for TSCRNN, MATEC, and SAM models are 8, 2, and 8, respectively. At a flow rate of 2000, the optimal batch sizes shift to 128, 128, and 32, respectively. These variations arise from differences in model parameters, which affect single-instance classification time. Across different models, *SmartTC* consistently achieves the shortest average classification time, outperforming fixed batch submission strategies.

The results demonstrate that *SmartTC* dynamically adapts to network load. Under low traffic, it accelerates task response by rapidly submitting small batches. Under high traffic, it optimizes throughput through efficient batch scheduling. This enables *SmartTC* to maintain stable and efficient traffic processing across varying load conditions.

Although the reduction in average classification time may appear minor, in practical scenarios such as intrusion detection or QoS-based routing, even second-level or millisecond-level differences can determine whether a system responds within the Service Level Agreement (SLA) window.

C. Host CPU Offloading

To quantitatively evaluate the resource-saving capability of *SmartTC*, we designed a comparative experiment based on the implementation mechanism of the FENXI system. Since FENXI requires dedicated CPU cores for polling-based packet processing (without the ability to share computational resources), we measured its resource consumption using the number of bound CPU cores. The experiment first deployed the FENXI system on a single-core CPU architecture, which was found unable to sustain full network interface throughput. Subsequently, we replicated a dual-core CPU system that achieved full network interface load while maintaining the

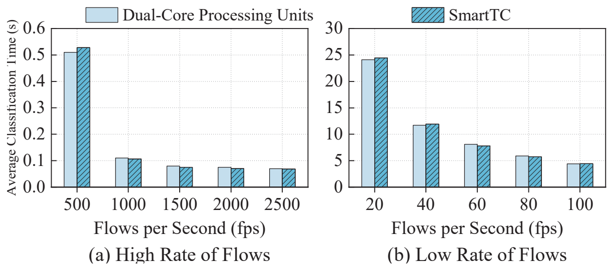

Fig. 10. The comparison between the *SmartTC* system and FENXI shows that even without the dedicated two CPU cores occupied by FENXI, *SmartTC* can obtain comparable or even better performance in terms of average classification time.

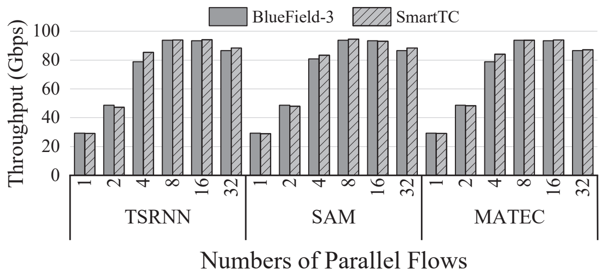

Fig. 11. The comparison of throughput between the *SmartTC* system and Bluefield-3 under varying numbers of parallel flows.

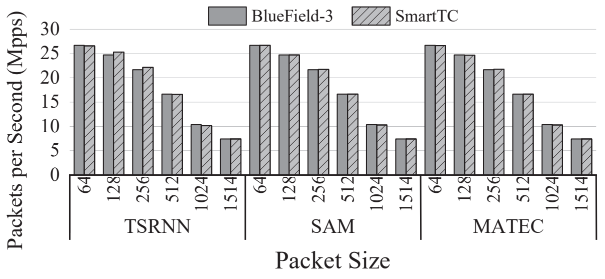

Fig. 12. The comparison of PPS between the *SmartTC* system with Bluefield-3 under varying packet sizes.

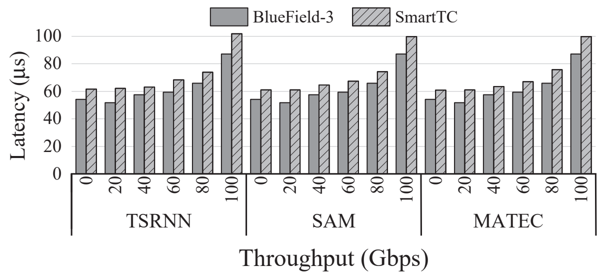

Fig. 13. The comparison of latency between the *SmartTC* system with Bluefield-3 under varying background traffic rates.

same average classification time as *SmartTC*, as shown in Figure 10.

The experimental results show that, compared to the FENXI system, which relies on two dedicated CPU cores for polling, *SmartTC* eliminates the need for host-side CPU resources but obtains comparable or even better performance in terms of average classification time. In a multi-tenant cloud environment, the resulting resource savings can free up valuable computational capacity for other critical tasks, without the side effect on the classification performance.

D. Throughput

The experimental results (as shown in the Figure 11) indicate that as the number of parallel flows increases, the throughput of *SmartTC* continues to rise and is nearly identical to the BlueField-3 system. When the number of parallel flows reaches 8, systems deploying TSCRNN, MATEC, and SAM can achieve 94.0 Gbps, 94.4 Gbps, and 93.7 Gbps, respectively, reaching nearly the full network card load throughput.

The experiment results demonstrate that *SmartTC* can maintain throughput close to the full capacity of the network interface, performing consistently with the BlueField-3 system without *SmartTC* deployed. This ensures that the traffic classification system does not affect the existing network throughput or degrade the QoS of other host tasks. Notably, when the number of parallel flows exceeds eight, the throughput slightly decreases for both BlueField-3 and *SmartTC*. This occurs because, as the number of parallel flows increases, the TCP congestion control mechanism limits the bandwidth of each flow, causing the overall throughput to initially rise with the increasing number of parallel flows and then decline to some extent. *SmartTC* can maintain stable system throughput in high-throughput environments, demonstrating that it has no adverse impact on the existing system under high load scenarios.

E. Packet Forwarding Capability

The experimental results (as shown in Figure 12) indicate that, in terms of packet forwarding capability, *SmartTC* performs similar or sometimes better than the BlueField-3 baseline system across different traffic classification models.

The results demonstrate that *SmartTC* does not become a performance bottleneck during packet forwarding. This is due to *SmartTC* employing an efficient pipeline parallel algorithm and fully utilizing the hardware resources of the SmartNIC. The forwarding efficiency of small packets depends on the host CPU, while large packets rely on the bandwidth of the PCIe channel. The introduction of *SmartTC* does not affect this forwarding capability, thereby ensuring stable packet forwarding throughput.

F. Latency

To evaluate the impact of *SmartTC* on latency under different traffic loads, we conducted latency tests under various background traffic throughput rates (0, 20, 40, 60, 80, 100 Gbps) to analyze the performance of *SmartTC* relative to the Bluefield-3 under different network loads.

The experimental results (as shown in the Figure 13) indicate that, as the throughput increases, both *SmartTC* and Bluefield-3 experience an increase in latency. Under no network load, *SmartTC*'s latency is 61.65 μs, which is an increase of 6.5 microseconds compared to the baseline system. When the network load reaches full card utilization, *SmartTC* 's latency rises to 101.74 μs, resulting in an additional 14.6 μs of delay compared to the baseline.

These results show that the introduction of *SmartTC* does indeed lead to some additional latency. However, even under full network load, the additional latency introduced by *SmartTC* is kept within approximately 10 μs. Since traffic analysis is an additional processing step, some increase in latency is inevitable. Traffic classification typically operates at the microsecond level, and the additional 10 μs delay does not significantly impact the classification results. Therefore, although *SmartTC* introduces some latency under high load, its effect on overall system performance is minimal, and it can maintain stable network performance without significantly affecting QoS.

V. Conclusion

This study proposes *SmartTC*, a real-time ML-based traffic classification system based on SmartNIC, which addresses resource constraints and dynamic network loads through hardware-software co-design, traffic-aware dynamic batch submission strategy, and parallel pipeline scheduling. By offloading analysis and pre-processing tasks to the SmartNIC and coordinating with host CPU and GPU resources, *SmartTC* reduces the average classification time by up to 90.9% under high loads and 16.8% under low loads compared to FENXI, while also reducing host CPU usage by at least two cores. In comparison to Bluefield-3, *SmartTC* maintains network performance. Experimental results demonstrate the feasibility of SmartNIC-based offloading, providing an efficient solution for machine learning-based traffic classification in cloud environments that reduces resource consumption and meets real-time performance requirements.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant Nos. 62432003, U22B2005, 92267206, and 62402487, as well as the LiaoNing Revitalization Talents Program under Grant No. XLYC2403086. Additional support is provided by the Basic Research Program of Shenzhen under Grant No. JCYJ20220531100804009, and the financial support from Lingnan University (LU) under Grant No. DB23A9.

References

[1] Jin Cheng, Yulei Wu, E Yuepeng, Junling You, Tong Li, Hui Li, and Jingguo Ge. Matec: A lightweight neural network for online encrypted traffic classification. *Computer Networks*, 199:108472, 2021.

[2] NVIDIA Corporation. Nvidia bluefield-3 networking platform, 2024.

[3] NVIDIA Corporation. Nvidia morpheus cybersecurity, 2024.

[4] Massimo Gallo, Alessandro Finamore, Gwendal Simon, and Dario Rossi. Real-time deep learning based traffic analytics. In *Proceedings of the SIGCOMM'20 Poster and Demo Sessions*, pages 76–78. 2020.

[5] Massimo Gallo, Alessandro Finamore, Gwendal Simon, and Dario Rossi. Fenxi: Deep-learning traffic analytics at the edge. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 202–213. IEEE, 2021.

[6] Yukai Huang, Lulu Chen, Chunpu Huang, Rui Zhang, Yiren Zhou, Ming Yan, and Jie Wu. Mitigating intra-host network congestion with smartnic. In *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2024.

[7] Intel. Receive-side scaling (rss), 2016.

[8] Syed Usman Jafri, Sanjay Rao, Vishal Shrivastav, and Mohit Tawarmalani. Leo: Online {ML-based} traffic classification at {Multi-Terabit} line rate. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1573–1591, 2024.

[9] Jongyul Kim, Insu Jang, Waleed Reda, Jaeseong Im, Marco Canini, Dejan Kostić, Youngjin Kwon, Simon Peter, and Emmett Witchel. Linefs: Efficient smartnic offload of a distributed file system with pipeline parallelism. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pages 756–771, 2021.

[10] Jong-Hyouk Lee and Kamal Singh. Switchtree: in-network computing and traffic analyses with random forests. *Neural Computing and Applications*, pages 1–12, 2020.

[11] Kunda Lin, Xiaolong Xu, and Honghao Gao. Tscrnn: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of iiot. *Computer Networks*, 190:107974, 2021.

[12] Xinjie Lin, Gang Xiong, Gaopeng Gou, Zhen Li, Junzheng Shi, and Jing Yu. Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification. In *Proceedings of the ACM Web Conference 2022*, pages 633–642, 2022.

[13] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing*, 24(3):1999–2012, 2020.

[14] Netronome. Netronome agiliotm cx 2x40gbe intelligent server adapter, 2018, 2018.

[15] Barefoot Networks. Intel® tofino™, 2021.

[16] Thuy TT Nguyen and Grenville Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE communications surveys & tutorials*, 10(4):56–76, 2008.

[17] Fannia Pacheco, Ernesto Exposito, Mathieu Gineste, Cedric Baudoin, and Jose Aguilar. Towards the deployment of machine learning solutions in network traffic classification: A systematic survey. *IEEE Communications Surveys & Tutorials*, 21(2):1988–2014, 2018.

[18] Ruixin Shi, Ming Yan, and Jie Wu. Optimizing inference quality with smartnic for recommendation system. In *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2024.

[19] Giuseppe Siracusano, Salvator Galea, Davide Sanvito, Mohammad Malekzadeh, Gianni Antichi, Paolo Costa, Hamed Haddadi, and Roberto Bifulco. Re-architecting traffic analysis with neural network interface cards. In *19th USENIX symposium on networked systems design and implementation (NSDI 22)*, pages 513–533, 2022.

[20] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1928–1943, 2018.

[21] Shangyi Sun, Rui Zhang, Ming Yan, and Jie Wu. Skv: A smartnic-offloaded distributed key-value store. In *2022 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–11. IEEE, 2022.

[22] Shivam Trivedi, Lauren Featherstun, Nathan DeMien, Callum Gunlach, Sagar Narayan, Jacob Sharp, Brian Werts, Lipu Wu, Carolyn Ellis, Lev Gorenstein, et al. Pulsar: Deploying network monitoring and intrusion detection for the science dmz. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*, pages 1–8. 2019.

[23] Xingda Wei, Rongxin Cheng, Yuhan Yang, Rong Chen, and Haibo Chen. Characterizing off-path {SmartNIC} for accelerating distributed systems. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, pages 987–1004, 2023.

[24] Dong Wen, Tao Li, Chenglong Li, Pengye Xia, Hui Yang, and Zhigang Sun. Octopus: A heterogeneous in-network computing accelerator enabling deep learning for network. *arXiv preprint arXiv:2308.11312*, 2023.

[25] Guorui Xie, Qing Li, and Yong Jiang. Self-attentive deep learning method for online traffic classification and its interpretability. *Computer Networks*, 196:108267, 2021.

[26] Guangmeng Zhou, Zhuotao Liu, Chuanpu Fu, Qi Li, and Ke Xu. An efficient design of intelligent network data plane. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6203–6220, 2023.