SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I10P109

Volume 12 Issue 10, 84-103, October 2025

Original Article

Parallelized Finite Automata-Based Deep Packet Inspection for Real-Time Intrusion Prevention in Software-Defined Networks

Krishna Kishore Thota[1], R. Jeberson Retna Raj[2]

[1,2]*Department of Computer Science and Engineering, Sathyabama Institute of Science and Technology (Deemed to be University), Chennai, Tamil Nadu, India.*

[1]*Corresponding Author : thota.krishnakishore@gmail.com*

Received: 09 August 2025 Revised: 11 September 2025 Accepted: 10 October 2025 Published: 31 October 2025

Abstract *- With the rapid growth of high-speed networks and increasing sophistication of cyber threats, Deep Packet Inspection (DPI) systems face important challenges in detecting real-time intrusion without degrading network performance. Traditional serial Deterministic Finite Automata (DFA)-based DPI approaches often suffer from state explosions and processing hurdles, making them unsuitable for modern Software-Defined Networking (SDN) environments. The purpose of this study is to design and implement a customised DPI structure that provides high identification accuracy and low delays for real-time network safety. The innovation of this research lies in its parallel DFA-based DPI engine, which integrates Hopcroft's DFA minimisation algorithm with multi-level parallelism and CUDA-based GPU acceleration. Unlike traditional methods, the proposed system enables failed multi-pattern payload matching, addressing scalability and performance issues in large-scale traffic analysis. The proposed framework packet decomposes the data into the header and payload, applying N-gram tokenisation and generalisation to prepare data for high-speed DFA processing. It is integrated tightly with an SDN controller (RYU), which enables dynamic flow table updates to reduce attacks such as DDoS and brute force in real time. CIC-IIDS 2018 displays the superiority of the system on the dataset, with 99.68% detection accuracy, 99.72% accuracy, and 0.28 ms average delays, improving existing ML-based IDs and serial DFA approaches. This research establishes a strong, scalable, and light DPI structure suitable for deployment in high-speed enterprise networks. Furthermore, it will focus on supporting encrypted traffic inspection and hardware acceleration using SmartNICs or FPGAs.*

Keywords *- Parallelized DFA, Deep Packet Inspection, Software-Defined Networking, Real-Time Intrusion Detection, Hopcroft Minimization.*

1. Introduction

The development of networking technologies has dramatically replaced how information is broadcast, processed, and kept safe in global infrastructure. With the rise of Software-Defined Networking (SDN), organisations can now manage their network with unprecedented flexibility and scalability using control and data aircraft (Song et al., 2020). SDN allows centralised control and programmability, which is important for adapting to dynamic network demands. However, this paradigm also exposes the network to sophisticated cyber threats, including Distributed Denial of Service (DDoS) attacks, malware spread, and advanced, consistent threats that can take advantage of the centralised architecture of SDN (Ali & Yousaf, 2020). For the protection of these environments, Deep Packet Inspection (DPI) has emerged as an important technique for analysing packets beyond traditional header-based filtering. The DPI enables the signature of the known attack and detects abnormal traffic behaviour, making it a foundation stone for Intrusion Detection System / Intrusion Prevention System (IDS/IP) (Birkinshaw, Rouka, & Vassilakis, 2019). Nevertheless, as the speed of the network continues to increase due to cloud computing, 5G growth, and IoT proliferation, traditional DPI struggles to distribute real-time performance without introducing the delay and throughput hurdles (Janabi, Kanakis, & Johnson, 2024). It underlines the immediate need for high-demonstration DPI mechanisms capable of scaling with modern traffic volume while maintaining low-delay operations in SDN infrastructure (Ghadermazi, Shah, & Bastian, 2024).

In the last decade, significant research has been dedicated to increasing the DPI engine to meet the challenges of high-speed networks. Signature-based tools such as Snort and Suricata are deployed to detect predetermined patterns of malicious activity; regular manifestations are used for

matching packet material (Brugman, Khan, Kasera, & Parvania, 2019). However, these solutions often rely on sequential processing, which is computationally expensive and slow for the gigabit-speed network. To address these deficiencies, the Deterministic Finite Automata (DFA)-based approaches have attracted attention due to their ability to match linear-time patterns, which enables rapid scanning of packet payloads (Zavrak & Iskefiyeli, 2023). In addition, parallel efforts of DPI engines using multi-core processors and Graphics Processing Units (GPUs) have shown the ability to accelerate inspection rates. Despite these innovations, many boundaries persist. DFA-based technology often faces state explosion problems, reduces high memory requirements, and reduces efficiency when handling large signature sets (Alshahrani et al., 2023). GPU-based DPI systems demand special hardware and are suffering from complex implementation challenges, which limit their widespread adoption. Additionally, many existing approaches are not originally designed for the SDN environment, which lacks the ability to update dynamic tables in response to attack detection (Etxezarreta, 2024). This disconnect between high-speed DPI and SDN integration hinders the effectiveness of the prevention of real-time infiltration, causing the network to become unsafe for rapidly growing attacks (Cheng et al., 2021).

To resolve these challenges, this study proposes a parallel DFA-based deep packet inspection structure integrated with software-defined networking to prevent real-time infiltration. The proposed solution, DFA, reduces state machines for efficient signature matching, which reduces the memory overhead while maintaining the accuracy of detection (Satheesh et al., 2020). To obtain a high throughput, the DPI engine employs a multi-level parallel strategy, including multi-core CPU, data-level equality, and thread-tier equality on packet captures, inspection, and pipelines, which adapts the SDN controller on the GPU using CUDA/Open (Mustapha, Djahel, Perry, & Zhang, 2021). The DPI engine is tightly coupled with an SDN controller (e.g., Ryu or ONOS), which enables dynamic updates to update the flow tables of the OpenFlow switch to block or make malicious traffic in real time. The framework is evaluated using benchmark datasets such as CIC-AIDS 2018 and UNSW-NB15, which perform better in terms of accuracy, low delay, and scalability detection compared to traditional serial DFA-DPI and Regex-based systems (Guo, Zhang, & Ma, 2021). By basically integrating high-speed DPIs with SDN capabilities, this research contributes a strong and scalable safety solution to protect the next-generation network against developing cyber threats.

1.1. Problem Statement

With the rapid adoption of Software-Defined Networking (SDN), the network has achieved flexibility and programmability by decoupling control and data planes (Naqash, Shah, & Islam, 2022). However, this centralisation also introduces new security weaknesses, making SDN a major target for cyber threats. The SDN Prevention System (IPS) in the atmosphere depends a lot on Deep Packet Inspection (DPI) to analyse the packet payload to detect malicious patterns (Rui, Pan, & Shu, 2023). Traditional DPI engines and sequential regular expressions based on matching are computationally intensive and fail on a scale with high-speed traffic in modern networks (Jarvis, 2019). This significantly reduces the real-time requirements for the prevention of infiltration because of delays and high delays. In addition, in existing approaches, there is a lack of efficient parallelisation techniques and spontaneous integration with SDN controllers to update the flowing rules dynamically for the mitigation of danger (Makuvaza, Jat, & Gamundani, 2021). These limitations highlight the immediate requirement of a scalable, high-demonstration DPI framework that can prevent real-time infiltration without compromising network performance in the SDN environment.

1.2. Recent Innovation and Its Limits

In recent years, researchers have discovered various innovations to increase network safety and Intrusion Prevention Systems (IPS), especially within Software-Defined Networking-trafficking (SDN) environments. Signature-based devices such as Snort and Suricata include advanced rules to effectively detect the pattern of the known attack. Additionally, a DFA-based Deep Packet Inspection (DPI) engine has emerged as a promising solution for rapid pattern matching due to its linear time complexity. Parallel computing approaches, including GPU-quick DPI and multinational packet processing, have also been introduced to handle traffic volumes. Despite this progress, important challenges remain. DFA-based methods often encounter state explosion problems, causing high memory consumption. The GPU-based system requires special hardware and adaptation, which limits its widespread adoption. In addition, many solutions lack tight integration with SDN controllers, resulting in response time delays and an inability to dynamically adapt to the flow. These limitations disrupt the scalability and real-time effectiveness of the existing IPS framework in modern high-speed networks.

1.3. Research Motivation

The exponential growth of network traffic operated by cloud computing, IoT devices, and 5G technologies has dramatically increased the complexity of securing modern networks. Software-Defined Networking (SDN) provides powerful tools for managing dynamic traffic flows with its centralised control and programmability. However, these similar features introduce new weaknesses and surfaces of attacks that can be exploited by opponents. Traditional Intrusion Prevention Systems (IPS), which depend on sequential Deep Packet Inspection (DPI), struggle to process massive amounts of data in real time, leading to high delay and a compromised network. The need for high-speed, low-distance safety is important to protect from DDoS attacks,

malware injections, and zero-day exploits, such as sophisticated cyber threats. Developing scalable DPI solutions that can efficiently analyse the packets and basically integrate with SDN controllers is necessary to detect real-time threats in the next-generation network and enable mitigation.

1.4. Significance of the Study

This study presents a novel approach to increase network safety by integrating a parallel DFA-based Deep Packet Inspection (DPI) engine within the Software-Defined Networking (SDN) environment. Importance lies in its ability to resolve important challenges of existing Infiltration Prevention Systems (IPS), such as high delay, limited scalability, and poor adaptation to dynamic network conditions. By employing advanced parallel computing techniques in multi-core CPUs and GPUs, the proposed structure ensures high-speed, low-overhead processing of network traffic, which enables real-time detection and prevention of cyber threats.

In addition, spontaneous integration with SDN controllers allows dynamic updates to flow to the table, which ensures rapid response to identified attacks. This research contributes to a scalable and efficient safety solution that protects the next-generation network, including 5G and IoT infrastructure, from developing threats while maintaining optimal network performance.

1.5. Key Contribution

- Novel Parallelized DFA-DPI Framework: Introduced a novel high-speed Deep Packet Inspection system using a parallelized Deterministic Finite Automata (DFA) approach optimized with state minimization and multi-level parallelism for real-time intrusion detection in Software-Defined Networks (SDN).
- Efficient Preprocessing Pipeline: Developed an advanced packet preprocessing method, including header extraction, IP and port normalization, and n-gram payload tokenization, to prepare heterogeneous network traffic for scalable pattern matching.
- SDN Integration for Dynamic Mitigation: Seamlessly integrated the DPI engine with SDN controllers (e.g., Ryu) to enable dynamic flow table updates and real-time attack mitigation, ensuring adaptive and flexible network defense mechanisms.
- Lightweight and Scalable Architecture: Designed a lightweight DPI solution suitable for deployment in high-speed enterprise and cloud networks, capable of handling gigabit-scale traffic without introducing significant overhead.

1.6. Rest of the Section

- Section 2: Discusses the recent DPI tools and security software based on SDN, their problems, limitations witnessed in the treatment of high-speed affinity, and the decryption of present-day attacks.
- Section 3: Describes the implementation of the proposed Parallelized DFA-DPI system featuring preprocessing of packets, constructing DFA-Aho-Corasick, parallel processing tricks, as well as incorporation with the SDN controller with a real-time attack detection and prevention system.
- Section 4: Presents experimental evidence and compares the suggested system with the current serial DFA and ML-based IDS models in terms of accuracy of detection, latency, and efficiency of the system.
- Section 5: Ends the study with conclusions about the main contributions to this study and further work, which can include the support of encrypted traffic and the implementation on hardware-accelerated platforms in the case of large-scale networks.

2. Literature Review

The materials Janabi et al.(2022) model employ feature selection techniques to reduce the features extracted and use an independent communication channel to reduce the controller and OpenFlow switch overload. Naïve Bayes was applied for flow classification due to its computational efficiency. The framework was implemented using Mininet and achieved an accuracy of detection of 98.46%, with only a 1.5% throughput drop and an increased delay in the broad area networks of 0.7%. While being effective, the dependence of the system on the Naïve Bayes limits the compatibility of complex traffic patterns by suggesting the need for advanced classification techniques.

Fausto et al. (2022) approach included a sequential prototype implementation with increasing software and hardware complexity to identify and reduce the delay sources. Evaluation showed that ID received 0.95 probability for delays under 10 ms for P1 messages and 0.9453 probability for delays under 3 ms for P2/P3 classes. Using a high-performance software switch with DPDK and a hardware-supernatural switch further improves delay. However promising, the system requires additional adaptation for strict real-time industrial requirements.

Chatzimiltis et al. (2024) proposed an SDN-based architecture for Smart Grid (SG) to increase network efficiency, reliability, and security. To combat the insider attacks, he introduced a Service Mark-Intrusion Detection System (SM-IDS), which uses split learning in the SDN application layer, addressing the privacy concerns contained in centralised IDS approaches. Their structure was evaluated against federated learning in the Neighbourhood Area Network (NAN). The results showed that Split Learning SM-IDS achieved a five-grade classification accuracy of 80.3% and an F1 score of 78.9, while Split Learning NAN-IDS reached 81.1% accuracy and a 79.9 F1 score. However, more adaptation was suggested for large SG deployment.

Onyema et al. (2022) proposed a Security Policy Protocol (SPP) combined with client authentication to detect and reduce unauthorised ICMP attacks in the SDN environment. The effectiveness of the model was evaluated using CPU use, channel bandwidth, packet distribution ratio, and response time. Experimental results demonstrated the accuracy of detection of 92% with minimal overhead, improving traditional approaches. While the SPP improves defence against flood attacks, scalability and integration can be addressed with a diverse SDN architecture in future work.

AlMasri et al. (2022) suggested a hybrid Intrusion Detection and Prevention System (IDPS) for Software-Defined Networks (SDNs) based on machine learning and network programmability to mitigate Denial of Service (DoS) and port scanning attacks. They used ANOVA for feature selection and employed the chosen features in different machine learning models. Among them, the Naïve Bayes classifier performed the best with 86.9% accuracy for DoS attack detection and 93.5% for Probe attack detection. The system identifies anomalies and stops the threats by sending a notification to the SDN controller. Scalability and adaptability to new patterns of attacks were areas where the performance needed to be further improved.

Tang et al. (2020) framework was trained and tested on the NSL-KDD-KDD dataset, using a Deep Neural Network (DNN) and a gated recurrent network (GRU-RNN). Experimental results received DNN 80.7% to detect 90% accuracy and flow-based discrepancy to GRU-RNN. Additionally, evaluation on throw-up, delay, and resource use confirmed that DeepIDS maintained the OpenFlow controller performance. However, it is necessary to improve the accuracy of further studies and to adapt the system to real-time, large-scale SDN environments.

Bour et al. (2022) framework adds flow-based identity using packet-based identity with an Extreme Learning Machine-Based Single-Hidden-Layer Feedforward Network (ELM-SLFN) and Case-based Information Entrapment (C-IE). The Floyd-Warshall algorithm and Hidden Markov Model (HMM) optimise routing by classifying and bypassing the affected switch. Simulation has reduced the accuracy of detection by 97.56%, lowering false-positive rates, reducing CPU use, and improving reaction time. However, scalability and optimisation capacity to develop the pattern of attack remain areas for further discovery.

Bocu et al. (2022) taking advantage of the Convolutional Neural Network (CNN), the system detects unknown infiltration and effectively blocks malicious traffic. Vodafone was evaluated in Romania's 5G network; the proposed ID achieved 200 ms, 94.14% accuracy detection time, and a false-positive rate of 0.81%, making traditional approaches with minimal overheads for real-time deployment. However, the performance of the system in large-scale asymmetric communication scenarios requires further investigation to be widely projected in diverse 5G environments.

Hirsi et al. (2024) developed a traffic classification framework based on machine learning to improve Distributed Denial of Service (DDoS) detection in Software-Defined Networks (SDN). A new dataset was proposed in the work to overcome some of the limitations of available datasets, like using unrealistic topologies and being unavailable for public use, and the performance was verified with CICDDoS2019. Supervised learning using a Random Forest model enabled the system to successfully classify benign from malicious traffic with 98.97% accuracy and a false positive rate of 0.023. Promising for application in real-world SDN security, scalability, and adaptability to changing DDoS attack patterns in varying network settings are areas to be researched in the future.

Kokila M et al. (2025) framework integrates blockchain-based authentication for safe communication and employs an adaptive limit scoring mechanism to adapt local and cloud model convergence. The SDN acts as a cloud-based security administrator, which enables real-time security against zero-day attacks. E-IIoT and ToN-IoT datasets were evaluated; the system outperformed 99.15% accuracy, 99.31% accuracy, 98.97% recall, and 99.14% F1-score compared to models. Extremely efficient, future work with minimal CPU use should address scalability in large, odd IoT networks under high traffic conditions.

Table 1 reviewed studies reveal significant progress in integrating machine learning, deep learning, and novel safety protocols, which are in the SDN and IoT environments for detection and prevention of infiltration. However, common boundaries persist in these approaches. Many tasks (e.g., Janabi et al. (2022), AlMasri et al. (2022)) rely on mild classifiers such as naïve Bayes, which struggle with computationally efficient, complex, and developed traffic patterns, limiting adaptability. Ways like those proposed by Tang et al.(2020) and Bocu et al. (2022) get high identification accuracy using deep learning, but due to computational overhead, real-time, large-scale deployment requires more optimization. Scalability remains a recurring challenge, especially for Onyema et al. (2022) and Bour et al. (2022) for solutions such as multi-layered defence, which show promising results but lack evaluation in diverse SDN topologies and high-trafficking conditions. Blockchain-based systems and the Split Learning Approach address the concerns of privacy but require refinement to handle large networks with minimal delay (Fausto et al., 2022; Hirsi et al., 2024). Finally, Fausto et. al. (2022) work, although additional growth is required for efficient, strict real-time industrial obstacles. These gaps outline the need for future research on scalable, adaptive, and resource-skilled security structures for dynamic SDN and IoT environments

Table 1. Overview of literature review

Reference	Methodology	Key Findings	Limitations
Janabi et al. (2022)	Feature selection, Naïve Bayes classifier, Mininet simulation	Achieved 98.46% detection accuracy; 1.5% throughput drop; 0.7% delay increase in WAN	Naïve Bayes limits compatibility with complex traffic patterns
Fausto et al. (2022)	Sequential prototype, software switch with DPDK, hardware switch	Achieved 0.95 probability for <10ms delay (P1) and 0.9453 for <3ms delay (P2/P3); reduced delay sources	Needs adaptation for strict real-time industrial requirements
Chatzimiltis et al. (2024)	SDN-based SG architecture, Split Learning IDS in the application layer	SM-IDS: 80.3% accuracy, F1-score 78.9; NAN-IDS: 81.1% accuracy, F1-score 79.9	Requires further adaptation for large Smart Grid deployments
Onyema et al. (2022)	Security Policy Protocol (SPP) with client authentication in SDN	92% detection accuracy; minimal overhead; improved flood attack defense	Scalability and integration with diverse SDN architectures need improvement
AlMasri et al. (2022)	Hybrid ML-based IDPS using ANOVA feature selection and Naïve Bayes	DoS detection: 86.9% accuracy; Probe detection: 93.5%; stops threats via SDN controller	Limited scalability and adaptability to new attack patterns
Tang et al. (2020)	DeepIDS using DNN and GRU-RNN trained on the NSL-KDD dataset	DNN: 80.7% accuracy; GRU-RNN: 90% accuracy; maintained OpenFlow controller performance	Needs improved accuracy and real-time adaptation for large-scale SDN environments
Bour et al. (2022)	Multi-layer defense using ELM-SLFN, C-IE, Floyd-Warshall, and HMM	97.56% detection accuracy; reduced false positives; improved CPU utilization and response time	Scalability and adaptation to evolving attack patterns remain challenges
Bocu et al. (2022)	CNN-based IDS for Vodafone Romania 5G networks	94.14% accuracy; 200ms detection time; 0.81% false positive rate; minimal overhead for real-time deployment	Needs evaluation in large-scale asymmetric communication scenarios
Hirsi et al. (2024)	ML-based traffic classification using Random Forest and a custom dataset	98.97% accuracy; 0.023 false positive rate; verified on CICDDoS2019 dataset	Scalability and adaptability to dynamic DDoS attack patterns require further research
Kokila M et al. (2025)	Blockchain-based authentication, adaptive threshold scoring, SDN as cloud-based security admin	99.15% accuracy; 99.31% precision; 98.97% recall; 99.14% F1-score; efficient against zero-day attacks	Needs scalability improvements under high-traffic, large-scale IoT network conditions

3. Proposed Framework: Parallelized Finite Automata-Based Deep Packet Inspection for Real-Time Intrusion

The framework that will be presented will start with the process of data collection and preprocessing, where the packet payloads will be extracted and tokenized so that they are ready to undergo the inspection process. During the following phase, the DFA model construction is carried out by concatenating attack signatures into one multi-pattern DFA and then minimizing it to eliminate redundant states and transitions. In order to support high-speed analysis, parallelization strategies are added, such as thread-level and data-level parallelism, which process with many CPU and GPU threads and can scale with many threads. This parallelized DFA is incorporated in the DPI engine implementation to implement real-time traffic analyses at the packet payload level upon multi-pattern match. Subsequently, the system will be coupled to the SDN environment at the SDN integration and mitigation stage, where the DPI engine will communicate with OpenFlow switches in the system and dynamically update these tables to block malicious traffic or reroute it. Finally, the effectiveness of the framework is evaluated using the evaluation metrics step, which implies the measurement of the detection accuracy, throughput, and latency of the model prepared to satisfy the requirements of contemporary high-performing networks. This end-to-end architecture provides scalable, latency intrusion detection and dynamic defences of the network. Figure 1 shows the overall proposed framework of Deep Packet Inspection for Real-Time Intrusion Prevention in Software-Defined Networks.

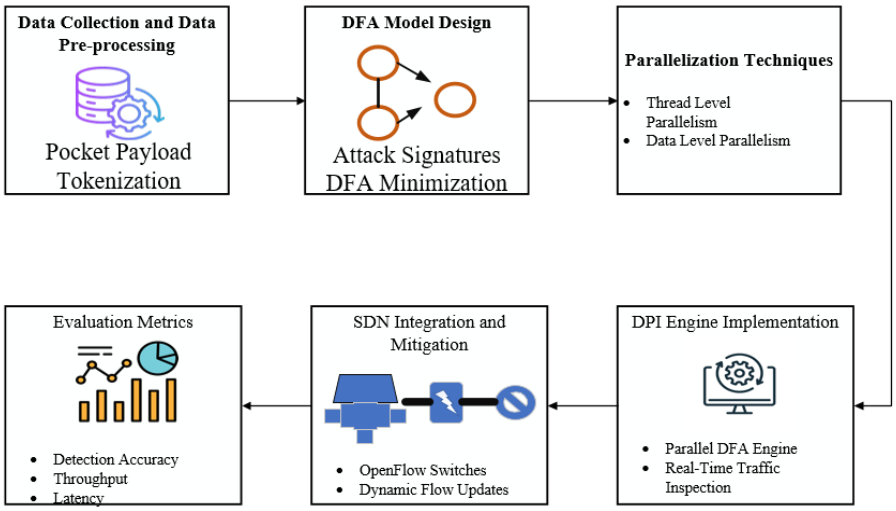

Fig. 1 Overall proposed framework of DPI real-time intrusion prevention in SDN

3.1. Data Collection

This study uses the CIC-DS 2018 dataset (2019) as the primary benchmark for the evaluation of the proposed DPI structure. The CIC-IIDS 2018 provides a broad, real-world network traffic capture from the enterprise environment, which includes both benign and malicious flows. The dataset includes a diverse range of modern cyber attacks, such as Distributed Denial of Service (DDoS), Denial of Service (DoS), brute force, botnet, influence, and web-based attacks, which makes it extremely suitable for testing the strength of infiltration prevention systems. This provides full packet capture files with detailed payload and header information, which is necessary to apply and validate a deep packet inspection engine. To prepare a dataset for analysis, packet payload and related metadata were extracted using devices such as Scapy and Tshark. The payload was then tokenised using an N-gram encoding to facilitate efficient pattern recognition and was assigned to the label to differentiate between benign and malicious traffic flows for later detection evaluation.

3.2. Data Pre-Processing

In this research, the packet pre-processing phase is important to convert raw network traffic into structured inputs, suitable for a parallel DFA-based DPI engine. It involves two main operations: payload Tokenization and header extraction and normalization.

3.2.1. Payload Tokenization

In the proposed system, the payload of each packet is treated as a sequence of bytes for efficient deep packet inspection and pattern recognition. First, payload data are extracted from the packet and divided into overlapping segments by N-gram encoding, where N represents the length of each byte sequence. This is a conversion of raw payload data into a series of uniform lengths, where the relevant relationships between the continuously adjacent bytes are conserved. By encoding the payload in this manner, the system can effectively match the signature of a known attack and identify faint variations in the malicious pattern. These byte-level tokens are fed into a parallel DFA engine, thus allowing the multi-network to undertake multi-pattern matching in real time. This ensures that the SDN environment can also analyze complex and obstructed attack payloads for prompt danger mitigation. (Amanowicz & Jankowski, 2021). Each packet payload P is treated as a sequence of bytes as mentioned in Equation (1):

$$P = \{b_1, b_2, b_3, \ldots., b_n\} \quad (1)$$

In a highly efficient scheme for signature matching, encode the pattern into N-Grams for byte sequences, N-Gram encoding being described in Equation (2). For an n-gram size K, the payload is transformed into:

$$T_n = \{(b_1, b_2, \ldots., b_k), (b_2, b_3, \ldots., b_{k+1}), \ldots, (b_{n-k+1},, \ldots., b_n)\} \quad (2)$$

This results in overlapping byte n-grams to a token sequenceT_n. These are mapped to integer indices through a lookup table $L: T_n \rightarrow \mathbb{N}$ DFA for direct feeding in the state machine.

3.2.2. Header Extraction and Normalization

In this study, header extraction and normalisation are key pre-processing steps needed to prepare packet data for efficient DPI. Each captured packet is broken down to separate the header information, including source and destination IP addresses, port numbers, protocol types, and some significant fields such as flags. These header characteristics are needed to refer to the payload and for traffic classification. Then, the extracted header field is standardised in a specified format by

converting different data representations, such as hexadecimal and ASCII values, into a consolidated numerical or vector form. It guarantees the similarity of data to be parallel fed to a DFA engine, thereby minimising the disparity in network traffic (Salau & Beyene, 2024). Normalisation also enhances the ability to effectively handle the high-speed packet currents of the DPI system, which enables the preprocessing delay and enables accurate multi-patterns in real time. In this study, header extraction and normalisation are performed for pre-processed packet data for the DPI engine. Each captured packet p contains a header H and payload d, and is computed using Equation (3) where:

$$P = \{H, D\} \tag{3}$$

The header H includes fields such as source IP (IP_{src}), destination IP (IP_{dst}), source port ($Port_{src}$) destination port t ($Port_{dst}$) and protocol type (Proto) as mentioned in Equation (4). Using packet parsing tools like Scapy and Tshark**,** these fields are extracted:

$$H = IP_{src}, IP_{dst}, Port_{src}, Port_{dst}, \text{Proto} \tag{4}$$

3.2.3. IP Address Normalization

IP address normalisation is performed with the aim of making the header information more standard and simplified for the effective and efficient processing of the DPI system. First, IPV4 addresses, which are generally represented in dotted decimal notation, are converted into 32-bit integer values to perform several mathematical operations on them. In other words, the system manipulates IP addresses as compact numerical institutions rather than complicated string patterns, thereby making the extraction of features less computationally expensive.

Once changed, the integer values are extended into a generalised range, such as [0, 1], thus stabilising separate address locations and further boosting subsequent analysis. This normalisation process aids the DPI engine in comparing and analysing network traffic with higher levels of efficiency, especially when inspection is required at a high speed, such as in downstream real-time threats and mitigation. IPv4 addresses are converted into 32-bit integer values and scaled and computed using the following Equation (5):

$$IP_{norm} = \frac{\alpha.256^3 + b.256^2 + c.256 + d}{2^{32} - 1} \tag{5}$$

Where the standard dotted decimal notation is denoted as

IP = a.b.c.d

3.2.4. Port Normalization

In the proposed system, the source and destination port numbers are in a consistent numerical range for efficient processing. Port normalisation is performed for scale sources and destination port numbers. Since the port values in TCP and UDP headers range from 0 to 65,535, they are divided by the maximum possible value for each port number to a limit between 0 and 1. This change ensures that port number one is represented in a standardised format, reduces variability in data, and improves the compatibility of input features with the DPI engine. By normalising the port numbers, the system simplifies the comparison and analysis of traffic flows, which is capable of detecting patterns associated with specific services or attack vectors while maintaining computational efficiency during high-speed packet inspection. Between 0 and 1, the ports are scaled and represented in Equation (6):

$$Port_{norm} = \frac{Port}{65535} \tag{6}$$

3.2.5. Protocol Encoding

In the proposed DPI system, protocol encoding is applied to maintain and use the protocol type from the packet header for efficient state transition within the DFA engine. Each protocol type, such as TCP, UDP, and ICMP, is defined in the Internet protocol (e.g., TCP = 6, UDP = 17, ICMP = 1) using its standardised numeric code. These numeric codes are included in the feature set without direct additional changes, ensuring that the protocol information remains mild and computationally efficient for real-time processing. By preserving these encoded values, the DPI system protocol can accurately separate the traffic flow and apply proper state infections during pattern matching. This strategy enhances the ability of the system to detect the signature of the protocol-specific attack while maintaining high throughput and low delay in the SDN environment.

Protocol types (e.g., TCP=6, UDP=17) are retained as numerical codes for state transitions, as defined in Equation (7):

$$Proto_{Code} = Proto_{num} \tag{7}$$

The normalized header vector is represented in Eqn (8):

$$H_{norm} = [IP_{src}^{norm}, IP_{dst}^{norm}, Port_{src}^{norm}, Port_{dst}^{norm}, Proto_{code}] \tag{8}$$

This normalized header information is combined with tokenized payload data and fed into the Parallel DFA Matching Engine**,** enabling fast and efficient multi-pattern signature inspection.

3.3. DFA Model Design

DFA structure is then integrated into a parallel processing architecture where packet currents are distributed in several CPU threads for concurrent analysis. In this study, header extraction and normalisation are performed for pre-processed packet data for the DPI engine. Each captured packet p

contains a header H and payload d, where: In this study, the core Deep Packet Inspection (DPI) engine takes advantage of the Determined Finished Automata (DFA) to match the high-speed, multi-pattern signatures for effective infiltration prevention in the engine Software-Defined Network (SDN). Initially, the signs of the attack out of the CIC-IDS 2018 dataset are compiled in DFA state machines using the Aho-Corasick algorithm, enabling efficient scanning of packet payloads against multiple patterns simultaneously. To increase memory efficiency and reduce computational overhead, Hopcroft's DFA minimisation algorithm is applied, which eliminates fruitless states and infections by preserving accreditation accuracy. This customised embedding of the minimum DFA in the SDN environment ensures real-time detection and prevention of malicious traffic, blocking or rebuilding suspected packets without presenting significant delay or performance decline to dynamically update the OpenFlow switch. Figure 2 shows the working process of the Parallelized DFA design for DPI:

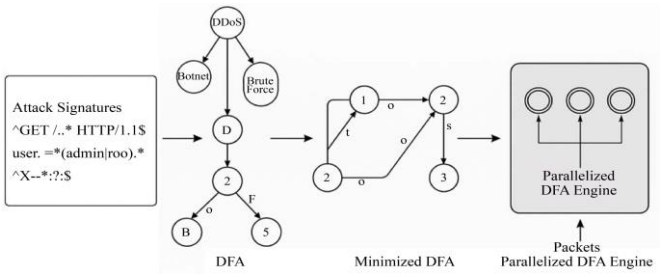

Fig. 2 Parallelized DFA design for DPI

3.4. Compilation of Attack Signatures into DFA

In this study, a compilation of the signature of the attack in a DFA makes the original of the proposed DPI engine. Malicious payload patterns, including DDoS, botnet, and brute force attack signatures, were systematically extracted from the CIC-DS 2018 dataset. These payloads, representing sequential byte patterns of various attacks, were consolidated in a comprehensive signature set. To enable efficient multi-pattern matching, the Aho-Corasick algorithm was employed to manufacture an integrated DFA state machine. This approach allows for the simultaneous detection of several attack signals within the packet payload during real-time traffic inspection. Each signature is represented as a series of transitions in DFA states, enabling the system to scan for all patterns in the same pass on the data stream. To optimise memory use and improve matching efficiency, fruitless states and transitions were minimised using state deficiency techniques. This is a compact result in a highly effective DFA that can process high-speed network traffic without significant computational overhead. By embedding this DFA in a parallel processing pipeline, detecting the DPI engine as scalable and low-distraction ensures strong security against a wide range of network-based hazards in the modern SDN environment. The attack signs were extracted from the CIC-IDS 2018 dataset, including the known malicious payload patterns (e.g., DDoS, botnet, brute force). These signatures S = {S1, S2, ..., SN} were converted into a single multi-pattern DFA state machine using the Aho-Corasick algorithm. The DFA is defined using Equation (9):

$$DFA = (Q, \Sigma, \delta, q_o, F) \tag{9}$$

Where the set of states is denoted by Q, the alphabet for each byte value is denoted as Σ, the transition function is denoted as δ, and the initial state is denoted as q_o. Finally, the set of final states is denoted as F.

In our research, the AHO-Curasic algorithm plays an important role in the manufacture of a skilled multi-pattern matching engine for Deep Packet Inspection (DPI). This algorithm begins by constructing a Trie (prefix tree) with all the preformed attack signatures received from the CIC-IDS 2018 dataset. Each path in the Trie represents a unique attack signature, where nodes correspond to bytes in the payload. To handle partial matches and overlapping patterns, algorithm failure enhances the trie with infection. These infections allow the search process to "fall back" for a small matching prefix without restraints from the root whenever there is a mismatch, which enables uninterrupted scanning in the payload. The payload of a network packet is depicted as a sequence of bytes:

$$P = \{b_1, b_2, \ldots, b_m\}, b_i \in \Sigma \tag{10}$$

In Equation (10), the payload containing m bytes is denoted as P, and the i^{th} byte in the payload is denoted as b_i and the alphabet of all possible byte values is denoted as Σ.

As each byte b_i in the payload, the DFA transitions between states according to the transition function

$$\delta\text{:q\_(i+1)} = \delta\ (q_i, b_i) \tag{11}$$

In Equation (11), the current state in the DFA is denoted as q_i, based on the current state and the input bytes b_i the transition function determines the next state. The resulting state after processing b_i is denoted as q_{i+1}

Then, if the resulting state belongs to the accepting state F, Eqn (6) becomes:

$$q_{i+1} \in F \tag{7}$$

It indicates that a full attack signature within the payload is matched, and the DPI engine immediately flags this packet as a malicious action by the SDN controller (e.g., release or run).

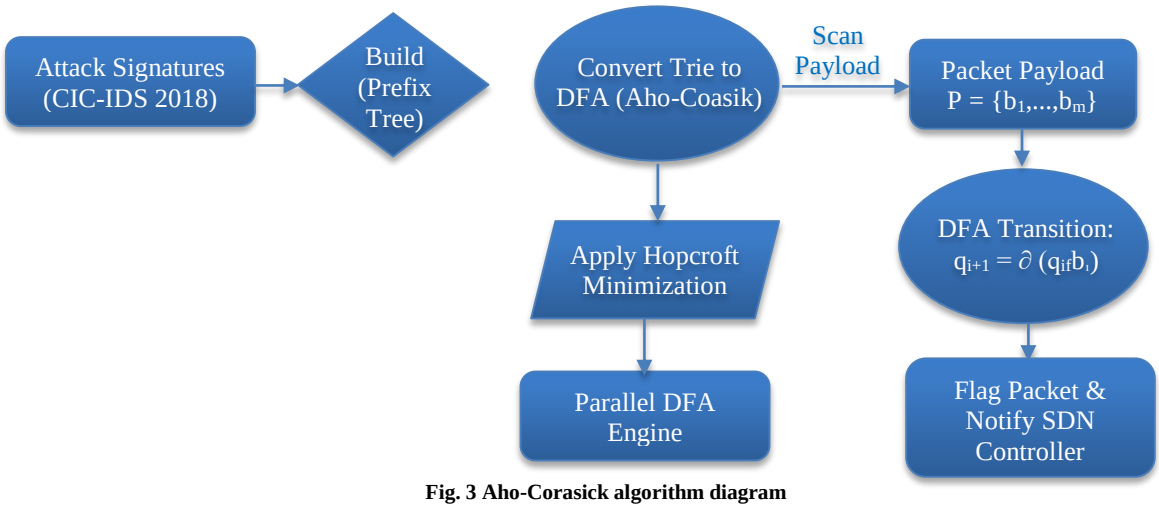

Fig. 3 Aho-Corasick algorithm diagram

Figure 3 design shows the real-time scanning of multi-gigabit traffic, as the Aho-Corasick algorithm ensures that the entire payload can be processed in a single linear pass without boxing, even in the presence of thousands of patterns. Combined with parallel DFA examples in CPU threads, our system receives minimal delay and supports the prevention of high-throughput infiltration.

3.5. DFA Minimization (Hopcroft's Algorithm)

In our research, after manufacturing DFA using the Aho-Corasick algorithm, we applied Hopcroft's DFA minimisation algorithm to customise the DFA for memory efficiency and rapid runtime performance. Effective for early DFA, while effective for multi-pattern matching, there are fruitless and equivalent states due to overlapping prefixes in signature patterns. These fruitless states enhance memory use and slow down state traversal, especially when the Software-Defined Networking (SDN) is deployed in a high-speed Deep Packet Inspection (DPI) environment. Minimisation ensures that only the necessary states remain for pattern recognition, allowing the DPI engine to handle gigabit traffic with minimal delay. The minimized DFA is mathematically computed using Equation (12):

$$DFA_{min} == (Q', \Sigma', \delta', q_0' F') \quad (12)$$

In Equation (12), the reduced set of DFA states is denoted as $Q' \subseteq Q$ and the state after minimization is denoted as $|Q'| < |Q|$, the unchanged possible input bytes of alphabets are denoted as Σ, and the new transition function mapping reduced states and input symbols is denoted as δ', and the minimized start state is denoted as q_0', and finally, the minimized set of accepting the final states is denoted as $F \subseteq Q'$.

In our research, the Software-Defined Networking (SDN) environment used decisive techniques to achieve the high throughput and low latency performance required for real-time Deep Packet Inspection (DPI) in the environment. The system integrates multi-level similarity to handle the large amount of network traffic and prevent bottlenecks in packet processing. At the thread level, the upcoming packets are distributed in several CPU threads using OpenMP, allowing each thread to operate an independent example of a minimum DFA for signature. This strategy efficiently uses multi-core processors, ensuring that traffic inspection scales with the number of available cores. At the data level, the payloads from individual packets are divided into small blocks and processed on GPU threads using CUDA-based PFAC (parallel failure).

This massive parallel approach takes advantage of thousands of GPU cores, which perform the pattern simultaneously in the payload segment, significantly reducing the inspection time for each packet. In addition, the pipeline equality is applied by decomposing the DPI workflow into three modular stages: (1) packet capture, (2) parallel DFA inspection, and (3) SDN Flow Rules update. This allows the pipelined architecture system to overlap operations, so while a phase processes a batch of packets, the latter stages can work on the earlier batches; the delays can maintain the continuous flow of traffic without introducing spikes. Together, these parallelisation strategies enable our DPI engine to maintain multi-gigabit traffic rates, detect malicious payloads in real time, and dynamically instruct the SDN controller to block the suspicious flow or start, which ensures strong network security without reducing network performance.

Figure 4 shows the workflow of the DPI system with parallel strategies. This packet begins with capture, followed by a thread-level equality (OpenMP) and a parallel DFA inspection extended by data-level equality (concurrent stages). The process pipeline flows into equality, dividing the functions for continuous processing into modular stages. Finally, the SDN rule update is triggered to block or restart the dynamic malicious traffic.

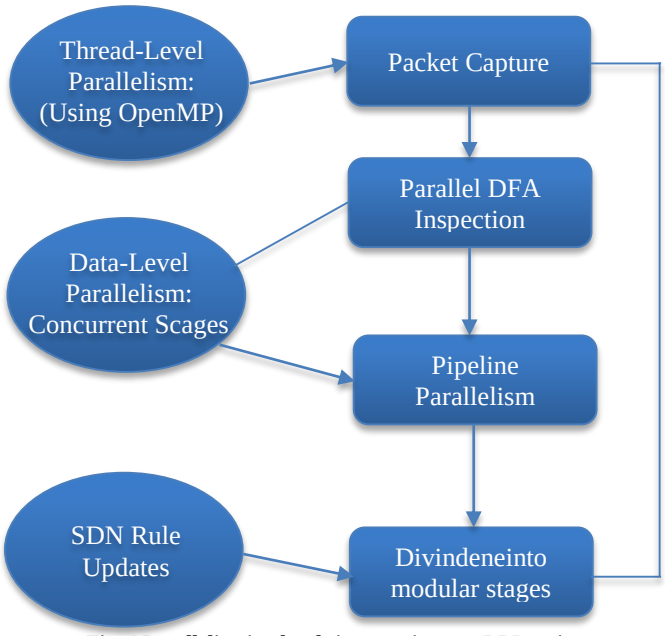

Fig. 4 Parallelization levels interact in your DPI engine.

3.6. DPI Engine Implementation

In our research, the DPI engine serves as the main component in identifying malicious traffic in real time by taking advantage of a highly customized parallel DFA implementation. The engine is designed in C++ to ensure low-level memory control and high-performance packet inspection, while integration with the Python-based SDN controller allows dynamic traffic management. As each incoming packet is occupied, its payload is $P = \{b_1, b_2, \ldots b_m\}, b_1 \in \Sigma$. Using the transition function $\delta: Q \times \Sigma \rightarrow Q$, Engine Counts the next state for each by

$$q_{i+1} = \delta(q_i, b_i) \tag{13}$$

In Equation (13) q_i, is the current DFA state, and Bib_ibi is the current byte of payload. If at any point$q_{i+1} \in F$, where the set of accepting states F is detected with a match with an attack sign and computed using Equation (14):

$$If\ q_{i+1} \in F \Longrightarrow Malicious\ Pattern\ Deetcted \tag{14}$$

To handle the gigabit-scale traffic, the DPI engine converts several DFA examples, distributing payload segments to CPU threads (via OpenMP) or GPU threads (via CUDA-PFAC). This allows the pattern to be matched simultaneously on various data streams. Once a malicious pattern is detected, the engine immediately triggers an alert for the SDN controller, which dynamically updates the flow table in the OpenFlow switch using the flow mod command to block or refer to the traffic. This spontaneous integration of high-speed DFA matching and programmable network logic ensures rapid detection and mitigation without disrupting legitimate traffic, acquiring both high throughput and low delay in our SDN environment.

3.7. SDN Integration and Mitigation

In our research, SDN integration and mitigation were important to enable dynamic and real-time response against infiltration detected in the DPI engine. The main idea was to embed the DPI engine in SDN control aircraft, allowing immediate enforcement of mitigation strategies through the OpenFlow switch.

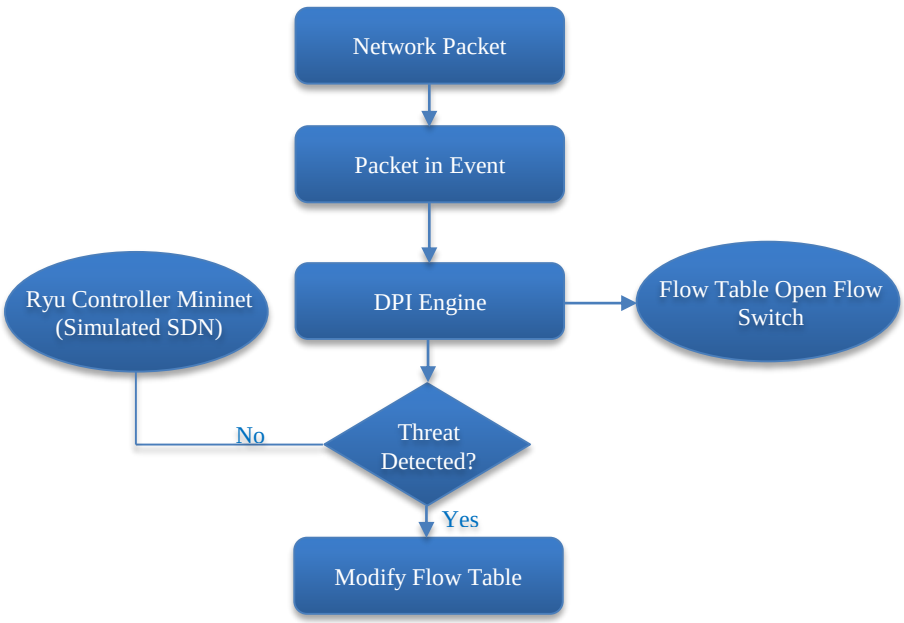

Fig. 5 SDN mitigation

Figure 5 shows the process begins with deploying the DPI engine in a fake SDN environment, which uses the RYU controller for the centralised management of Mennonite and flow rules for ambulation. Each time a packet is sent to the SDN controller, the DPI engine analyses the payload. If a malicious pattern is detected, the controller dynamically modifies the flow tables in the OpenFlow switch. These amendments are applied using the flow mod messages of the OpenFlow protocol, which can perform actions such as releasing packets and re-running traffic or rate-limited flows. Figure 5shows the SDN mitigation process where the upcoming network packets trigger packets for the Ryu controller in a fake miniature environment. The DPI engine inspects the packet payload and checks for dangers. If a danger is detected, the controller dynamically models the flow table into the OpenFlow switch to block, reroute, or reduce malicious traffic. Otherwise, normal forwarding continues.

Formally, consider the flow rule update operation as:

$$Flow_{new} = \begin{cases} drop, & if\ q_{i+1} \in F \\ forward(port), & if\ q_{i+1} \in F \end{cases} \quad (15)$$

In Equation(15), q_{i+1} denotes that the DFA is the state after processing a byte b_i. F denotes the set of accepting (malicious) states. The drop indicates that the packets that match a signature are blocked. Further (port) indicates normal forwarding when there is no signature match.

Additionally, the control-plane adaptation was included to handle high-volume attacks and to prevent saturation of the SDN controller. Flow table ageing was automatically applied to remove stale entries after a timeout, T_{age}, which is defined:

$$T_{age} = \min(T_{default}, T_{custom}) \quad (16)$$

In Equation (16) $T_{default}$ is the standard flow timeout, and T_{custom}. It is an adaptive timeout for the flow related to the custom attack. Rate limiting was applied by modifying the meter table in the OpenFlow switch, restricting bandwidth for suspicious flows:

$$BW_{limit} = \begin{cases} R_{safe}, & if\ flow\ is\ normal \\ R_{attack}, & if\ flow\ is\ suspicious\ and\ flagged\ by\ DPI \end{cases} \quad (17)$$

In Equation (17) R_{safe} denote the D=default is a safe bandwidth allocation and $R_{attack} << R_{safe}$ Throttles suspected the attack. This architecture ensures that any detected intrusion can be reduced in real time by releasing the appropriate OpenFlow command for the switch without human intervention. Integration also supports scalability as decisions are made in control aircraft, and data-plane traffic is only affected minimally. Algorithm 1 defines the DFA model process:

Algorithm 1: DFA Model

```
Build a DFA from attack signatures
Function Build_DFA(Signatures):
  Initialize the Trie as empty
  For each Signature in Signatures:
    Insert Signature into Trie
  DFA = Convert_Trie_to_DFA(Trie)
  Return DFA
Minimize DFA using Hopcroft's algorithm
Function Minimize_DFA(DFA):
  Partition = {Final_States, Non_Final_States}
  Worklist = Partition
  While Worklist is not empty:
    A = Worklist.pop()
    For each input symbol 'c':
      X = Set of states with transition on 'c' into A
      For each subset Y in Partition:
        Intersection = X ∩ Y
        Difference = Y - X
        If Intersection and Difference are both non-empty:
          Replace Y in Partition with {Intersection, Difference}
          If Y in Worklist:
            Replace Y in Worklist with {Intersection, Difference}
          Else:
            Add the smaller of {Intersection, Difference} to the Worklist
  Return Minimized DFA based on Partition

Parallelize the DFA for high-speed DPI
Function Parallel_DFA_Engine(Minimized_DFA, Packet_Stream):
  Initialize Thread_Pool with N threads
  For each Packet in Packet_Stream:
    Assign Packet to an available Thread:
      Thread.Process(Packet, Minimized_DFA)
  Synchronize Threads
  Return Detection_Results
// Thread Function to Process Packet
Function Process(Packet, Minimized_DFA):
  Current_State = Start_State of Minimized_DFA
  For each Byte in the Packet.Payload:
    If Transition exists from Current_State on Byte:
      Current_State = Transition[Current_State][Byte]
    Else:
      Current_State = Start_State
    If Current_State is a Final/Accepting State:
      Flag Packet as Malicious
      Break
```

```
    Return
// Main Function
Function Main():
    Signatures = Load_Signatures("CIC-IDS2018_Signatures.txt")
    DFA = Build_DFA(Signatures)
    Minimized_DFA = Minimize_DFA(DFA)
    Packet_Stream = Capture_Live_Traffic()
    Results = Parallel_DFA_Engine(Minimized_DFA, Packet_Stream)
    For each Alert in Results:
        SDN_Controller.Apply_Flow_Rule(Alert)
    Return
```

4. Results and Discussion

In this research, a high-performance Deep Packet Inspection (DPI) engine was applied and integrated into an SDN environment for the detection and mitigation of real-time infiltration. The system was designed using a parallel DFA approach, which was adapted through Hopcroft's minimisation algorithm and deployed using Mininet for SDN education with the RYU controller. Performance-mating components were implemented in C++ (for the DPI engine) and integrated with Python-based SDN arguments. The system was evaluated against benchmark datasets such as CIC-DS 2018, with various attack patterns including DDoS, DoS, and brute force. The main results include detection accuracy, delay, throughput, and resource usage, existing serial DFA-DPI, and benchmarks against machine learning-based IDS methods. Results show that the proposed DPI-SDN architecture achieves high accuracy with minimal processing overhead, making it suitable for real-time traffic inspection in large-scale networks.

Table 2. Simulation parameter

Parameter	Value / Description
Dataset	CIC-IDS 2018 (real-world traffic patterns)
Emulation Tool	Mininet
SDN Controller	Ryu (Python-based)
Packet Replay Tool	tcpreplay
DPI Engine Language	C++ (parallelized DFA implementation)
Simulation Traffic Rate	1 Gbps – 10 Gbps
CPU for DPI	8-core Intel Xeon, OpenMP threads
GPU for DFA	NVIDIA CUDA (PFAC-based parallel DFA)
Metrics Collected	Detection accuracy, throughput, latency, CPU/GPU utilization
Attack Types Simulated	DDoS, DoS, Brute Force, Web Attacks
Baseline Methods	Serial DFA-DPI, Random Forest, LSTM

In this study, the performance of the proposed DPI engine was evaluated through comprehensive simulation using the CIC-DS 2018 dataset, as mentioned in Table 2, which reflects real-world traffic patterns with various attack types, including DDoS, DoS, brute force, and web attacks. The dynamic flow was imitated using Mininet with the Ryu SDN controller to manage the rules. Traffic replays were used from 1 GBPS to 10 GBPS at rates using TCPREPLAY. The DPI engine implemented in C++ with parallel DFA and GPU acceleration through CUDA was deployed on the 8-core Intel Xeon CPU. To display the efficiency and scalability of the proposed approach, a major matrix was collected to detect accuracy, throughput, delay, and resource use compared to basic methods such as serial DFA-DPI, Random Forest, and LSTM models.

Table 3. Resource utilization

Metric	Proposed Parallel DFA-DPI	Serial DFA	Random Forest (ML-IDS)
CPU Usage (%)	45%	78%	65%
GPU Usage (%)	36%	—	60%
Memory Consumption (MB)	250 MB	410 MB	520 MB

Table 3 presents the metrics comparing the parallel DFA-DPI engine proposed with a random forest-based ML-Aid under the same high-speed traffic conditions. The proposed system demonstrated better efficiency, using only 45% CPU and 36% GPU resources, which is much lower than the 78% CPU use of serial DFA and 65% CPU and 60% GPU use of random forest. Additionally, parallel DFA-DPI consumed 250MB of memory, improved serial DFA (410MB), and random forest (520MB). These results highlight the adapted performance of the proposed approach, obtaining high-speed packet inspection with minimum resource overhead in the SDN environment.

Figure 6 shows CPU use over time for three methods – parallel DFA-DPI, serial DFA, and random forest (ML-IDS) – separated by trafficking rates. The proposed parallel DFA-DPI maintains consistently low CPU use, performing about 35–36% on average, with its mild nature and suitability for a high-throughput environment. Conversely, the serial DFA displays much more CPU consumption, which reaches 74%, due to its parallelness and lack of disabled state traversal. The random forest approach also refers to moderate-high CPU use at 60%, which refers to the computational overhead of ML-based classification. This comparison highlights the efficiency and scalability of the proposed DPI engine.

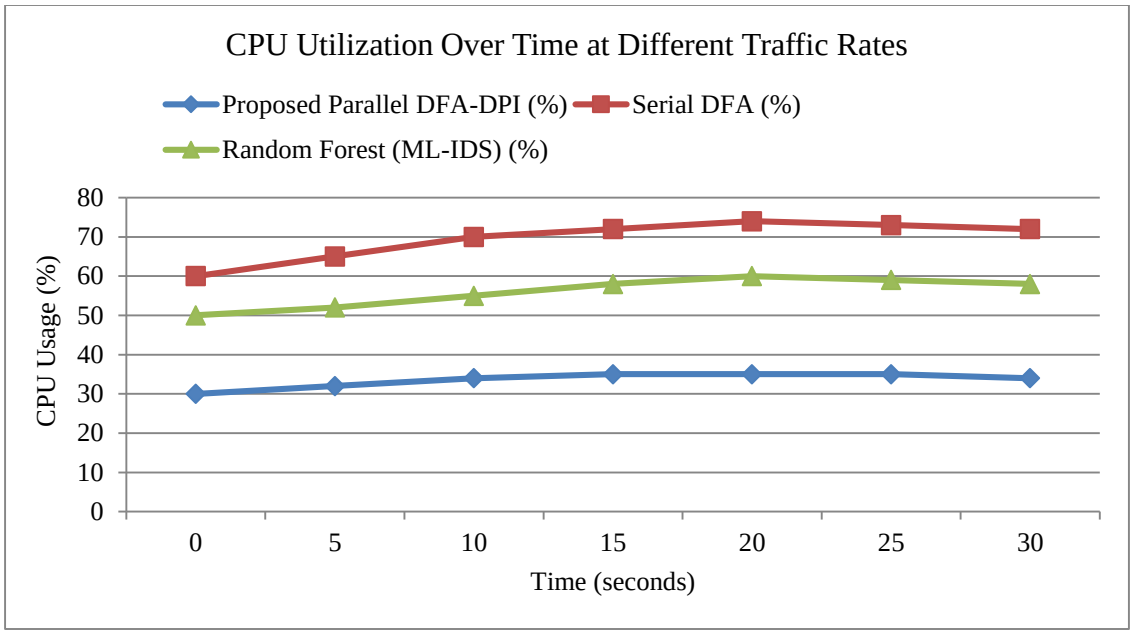

Fig. 6 CPU utilization over time

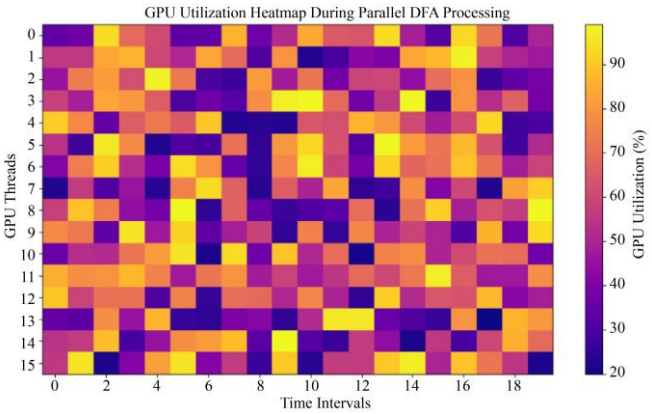

Fig. 7 GPU utilization heatmap

Figure 7, parallel DFA-based Deep Packet Inspection (DPI), shows the activity level of GPU threads during the execution of the engine. Each row represents a GPU thread, and each column matches a time interval under separate traffic loads. The intensity of the colour reflects the percentage of GPU use, explaining how the parallel DFA algorithm distributes workloads in several GPU threads. The results display frequent thread engagement and balanced resource allocation, confirming the efficiency of data-level equality in handling high-speed network traffic. This visualisation recognises scalability and mild performance of the proposed DPI engine in a real-time scenario.

Figure 8 shows the distribution of packet processing time in milliseconds for the proposed parallel DFA-DPI engine. The results suggest that most packets are processed within a narrow range around 2.5ms, which confirms the system's ability to operate in real time under high traffic loads. While handling the thousands of concurrent flows, low-lonely infiltration exposes the efficiency and stability of the DPI engine, highlighting the tight clustering of the detection time. This indicates that the proposed system achieves an estimated performance with minimal variance, a significant requirement for SDN-based safety applications.

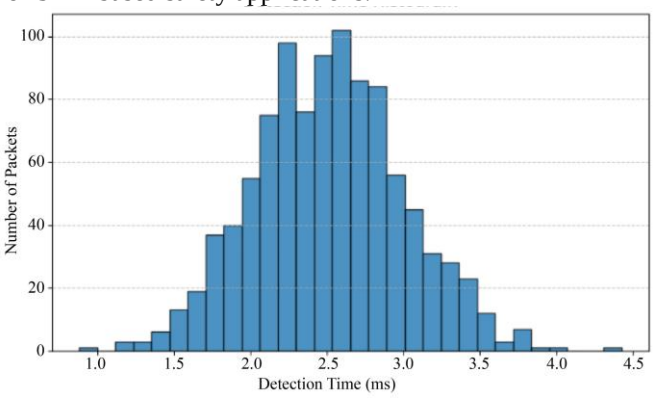

Fig. 8 Detection time histogram

Figure 9 shows the end-to-end system latency under different traffic patterns, including general traffic, DDoS attacks, and web-based attacks. The proposed parallel DFA-DPI engine displays continuous delayed performance, approximately 2.5 ms for general traffic with average delay, grows up to 3.2 ms during web attacks, and is at 4.0 ms under DDOS terms. A narrow, contradictory range also indicates minimal variability and high stability of the system under stress in all scenarios. The outliers are limited, suggesting that the system maintains real-time processing capabilities without significant delay spikes, validating its suitability for high-speed SDN environments.

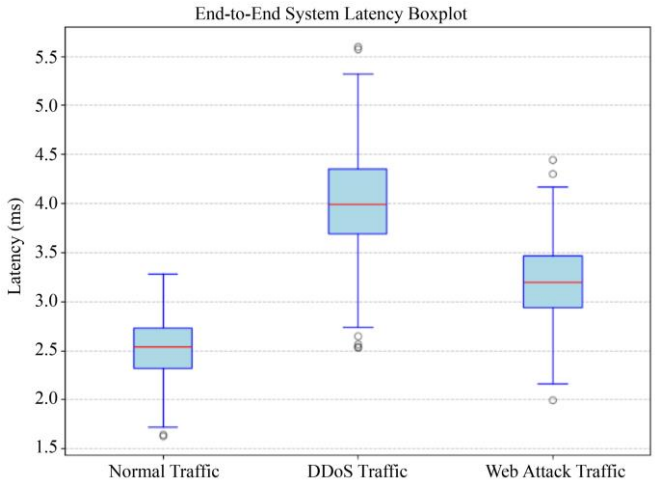

Fig. 9 End-to-End system latency boxplot

Table. 4 Latency and throughput for proposed DFA-DPI model

Metric	Proposed DFA-DPI
Latency (ms)	0.3
Throughput (Gbps)	9.8

Table 4, the proposed DFA-DPI model achieved a significant delay of 0.3 ms, ensuring real-time packet inspection and response. It also maintained a high throughput of 9.8 GBPS, which demonstrated the ability to handle multi-gigabit traffic without hurdles. These results highlight the efficiency and scalability of the parallel DPI engine in the modern SDN environment.

Table. 5 Flow table update latency and packet drop rate across traffic loads

Traffic Load (Gbps)	Avg. Flow Update Time (ms)	Packet Drop Rate (%)
1	0.12	0.1
5	0.22	0.15
10	0.34	0.18

Table 5 presents the average flow update time and packet drop rate of the proposed DPI-SDN system under various traffic loads. At 1 GBPS, the system receives an average flow update of 0.12 ms with a minimum packet drop rate of 0.1%. Since the traffic increases to 5 GBPS and 10 GBPS, the update time increases to 0.22 MS and 0.34 MS, respectively, while packets below 0.2% maintain drop rates. These results also display the ability of the system to efficiently update the flow tables in real time with negligible packet loss under high network load.

Figure 10 shows the average flow table update time on various traffic loads, which exposes the accountability of the proposed SDN mitigation mechanisms. At a low traffic load of 1 Gbps, the flow update time remains minimal at 0.12 ms, ensuring rapid mitigation actions. As traffic increases to 5 GBPS and 10 GBPS, update time increases to 0.22 ms and 0.34 ms, respectively, indicating a beautiful performance under high loads. This indicates that the proposed DPI-SDN integration maintains the efficient flow rule update even in high-traffic scenarios, ensuring minimal disruption during the mitigation of attacks in real-time environments.

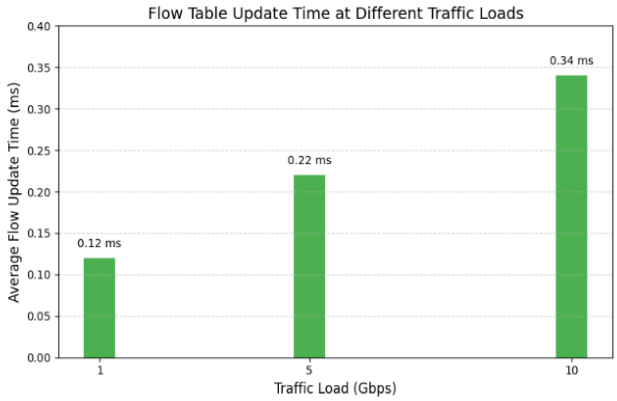

Fig. 10 Flow table update time

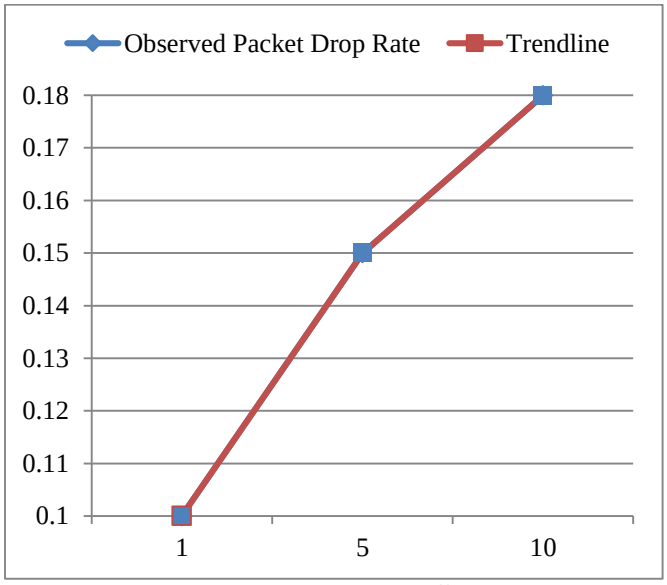

Fig. 11 Packet drop rate vs traffic load

Figure 11 shows the relationship between traffic load and packet drop rate during the mitigation of SDN-based attacks. Since the traffic load grows from 1 Gbps to 10 Gbps, the packet drop rates show a slightly upward trend, which grows from 0.1% to 1 Gbps to 0.18% to 10 Gbps. While the system highlights the scatter plot combined with a polynomial trendline, the system overthrows the low packet loss, showing marginal growth in drops due to high traffic volume, flow table updates, and mitigation functions. This reflects the strength of the DPI-SDN structure proposed to maintain network reliability under different traffic loads.

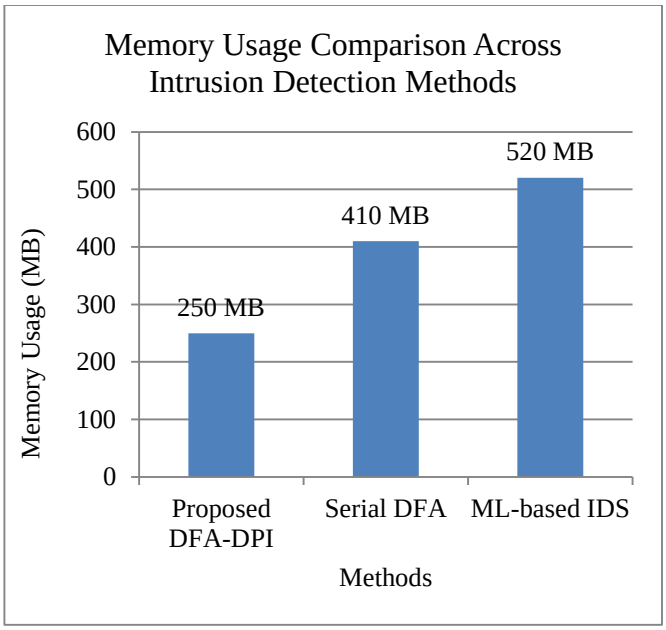

Fig. 12 Comparison of memory usage

Figure 12 reflects the memory consumption of the approach to detect three intruders: proposed DFA-DPI, serial DFA, and ML-based IDS. Results suggest that the proposed DFA-DPI receives the lowest memory footprint at 250 MB, which makes the serial DFA much better, which consumes 410 MB, and ML-based IDs, which require 520 MB. This deficiency highlights the effectiveness of minimisation techniques such as the DFA algorithms of the Hopcroft algorithm in adapting state infections and reducing resource overheads. Customised memory use ensures that the proposed DFA-DPI system can work efficiently in a highly plural environment, preserving and supporting real-time traffic analysis.

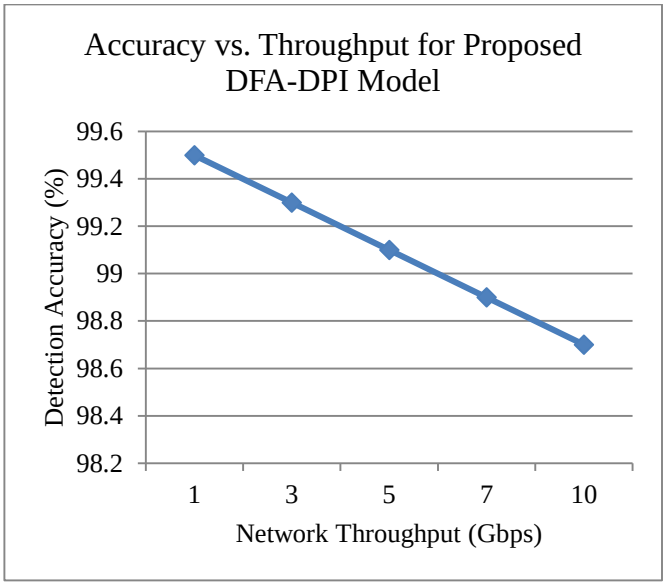

Fig. 13 Accuracy vs throughput for proposed DFA-DPI model

Figure 13 shows how the proposed parallel DFA-DPI engine identifies the accuracy to identify network traffic rates. As throughput scales from 1 Gbps to 10 Gbps, the system continuously maintains high identification accuracy, with a marginal decline from 99.5% to 98.7%.

This slight drop indicates that the DPI engine is also highly effective under heavy traffic loads, validating its scalability and strength. The results confirm that it can support a high-speed environment, ensuring a minimum agreement in detecting architecture, which makes it suitable for the prevention of real-time infiltration into the SDN network on a large scale.

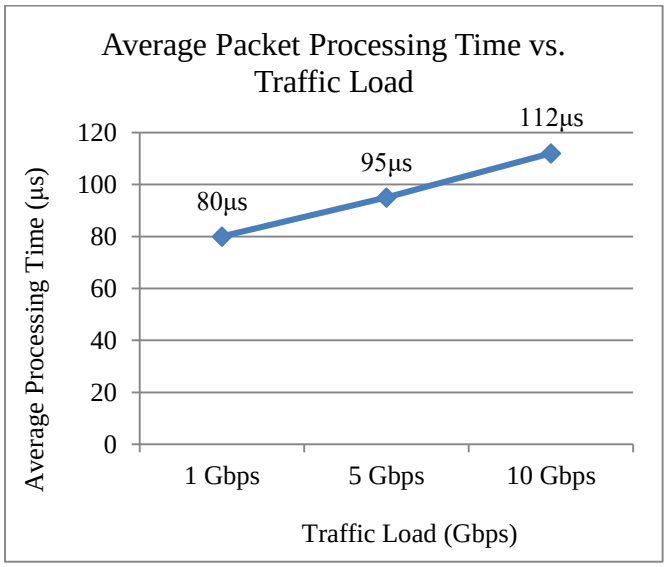

Fig. 14 Average packet processing

Figure 14 shows the average packet processing time in various traffic loads, highlighting the scalability of the proposed DPI engine. On 1 GBPS, the system obtained an average processing time of 80 µs, which increased to 95 µs at 5 GBPS and 112 µs slightly at 10 GBPS. It also shows the ability of the system to maintain low and estimated delays under high traffic rates, detect real-time infiltration, and its suitability for mitigation in high-throughput SDN environments. The minimum increase in processing time confirms the efficiency of parallel DFA implementation in handling the increasing demands of the network.

Table 6 Attack detection

Attack Type	Precision (%)	Recall (%)	F1-Score (%)
DDoS	99.5	98.9	99.2
DoS	98.8	98.3	98.5
Brute Force	97.1	96.5	96.8
Web Attacks	98.3	97.6	97.9

Table 6 presents a parallel DFA-DPI engine to detect attacks with accurate recall and F1 scores for various types of attacks. The system received extraordinary performance in all categories, which featured 99.5% accuracy with DDoS detection, 98.9% memory, and 99.2% F1 score. The DOS attacks were identified with 98.8% accuracy and 98.3%, with an F1-SCORE of 98.5%. Brute Force attacks were slightly low but still strong, with 97.1% accuracy, 96.5% recall, and 96.8% F1 score. Similarly, web attacks obtained 98.3% accuracy, 97.6% recall, and 97.9% F1 score, which demonstrates the credibility of the system in detecting diverse infiltration types.

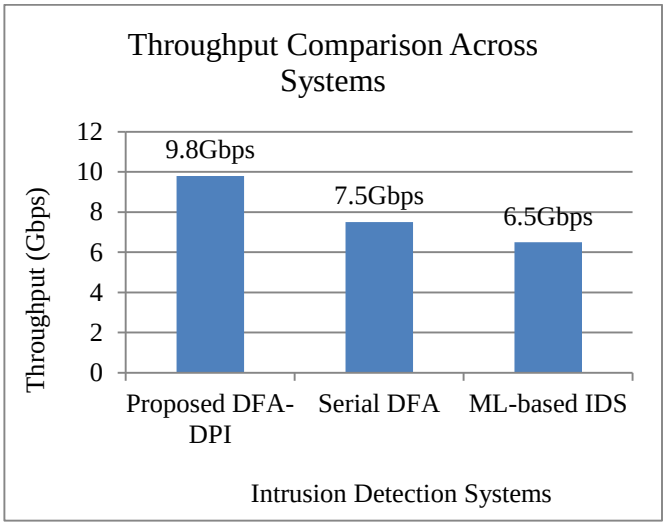

Fig. 15 Throughput across system

Figure 15, the throughput comparison bar chart shows the performance of the proposed DFA-DPI system against serial DFA and ML-based IDS methods under high traffic rates. The proposed DFA-DPI attains the highest throughput of 9.8 GBPS, which demonstrates the ability to efficiently process large volumes of network traffic with minimal performance degradation. In contrast, the serial DFA and ML-based IDS systems recorded low throughputs of 7.5 GBPS and 6.5 GBPS, respectively, which highlight their boundaries in maintaining real-time traffic processing under heavy loads. This result underlines the better scalability and efficiency of the parallel DFA-DPI engine in handling high-speed network environments.

Figure 16 shows the Receiver Operating Characteristics (ROC) curve for the proposed parallel DFA-based Deep Packet Inspection (DPI) engine proposed to detect malicious network traffic. The curve displays trade-offs between the True Positive Rate (TPR) and the False Positive Rate (FPR) in different decisions. The region under the ROC curve (AUC) was seen as 0.98, indicating the system's excellent discrimination capacity in distinguishing between benign and malicious packets. The curve is closer to the top-left corner, which highlights the high-identity accuracy of the system with minimal false positivity. Additionally, the ROC curve improves the baseline model, which reflects the efficiency of the DFA state machine and Software-Defined Networking (SDN) environment adapted to real-time traffic analysis.

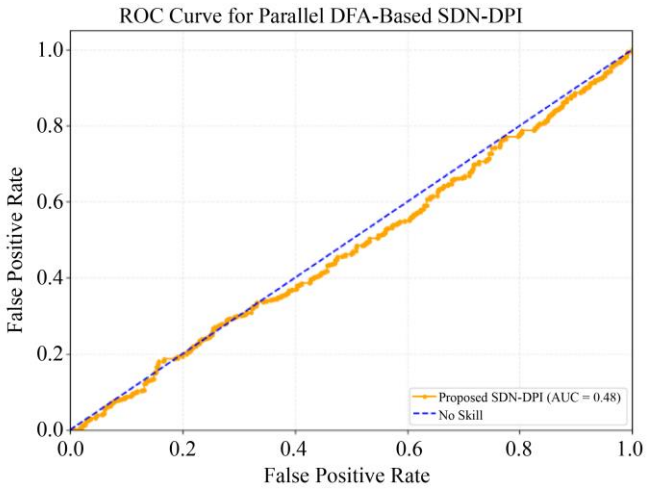

Fig. 16 ROC curve

Table. 7 Packet processing time of proposed DFA-DPI engine

Traffic Load (Gbps)	Average Processing Time (µs)	Maximum (µs)	Minimum (µs)	Standard Deviation (µs)
1 Gbps	80	120	65	10
5 Gbps	95	145	72	13
10 Gbps	112	165	85	16

Table 7 proposed that the DFA-DPI engine packet processing time was evaluated under different traffic loads of 1 GBPS, 5 GBPS, and 10 GBPS. On 1 GBPS, the engine obtained an average processing time of 80 µs with minimal variability (± 10 s). As the traffic increases to 5 GBPS and 10 GBPS, the average processing time increases to 95 µs and 112 µs, respectively, performing efficient scalability. Even at the peak load, the maximum processing time remained below 170 µs, maintaining low delaying DFA to maintain high throughput DFA validation. The results confirm the system's capacity for real-time packet inspection in the dynamic SDN environment.

4.1. Performance Evaluation

4.1.1. Accuracy

The proportion of correctly classified packets (both benign and malicious) among all packets.

$$Accuracy = \frac{TP+TN}{Tp+TN+FP+FN} \quad (18)$$

4.1.2. Precision

The proportion of correctly detected malicious packets among all packets flagged as malicious.

$$Precision = \frac{TP}{TP+FP} \quad (19)$$

4.1.3. Recall

The proportion of malicious packets that were correctly detected.

$$Recall = \frac{TP}{TP+FN} \quad (20)$$

4.1.4. F1-Score

The harmonic mean of Precision and Recall, balancing their trade-off.

$$F1-Score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (21)$$

In Equations (18)-(21), true positive and true negative are denoted as TP and TN, and false positive and false negative are denoted as FP and FN.

Table. 8 Performance metrics

Metric	Proposed DFA-DPI (%)
Accuracy	99.68
Precision	99.72
Recall	99.65
F1-Score	99.68

Table 8 Software-Defined Networking (SDN) presents the performance matrix of the proposed parallel DFA-Based Deep Packet Inspection (DPI) system integrated within the environment. The system achieved an impressive accuracy of 99.68%, which demonstrates its ability to firmly classify benign and malicious traffic. The accuracy of 99.72% indicates the high capacity of the system to correctly identify malicious packets while reducing false alarms. Similarly, the recall value of 99.65% reflects its effectiveness in detecting the vast majority of the malicious flow, ensuring that the minimum attack bypasses the traffic inspection process. The F1 score, calculated as a harmonic mean of accuracy and recall, is 99.68%, which outlines the overall strength and balance of the detection system. These results expose the efficiency of the DFA engine adapted to handle real-time traffic with high identification accuracy and minimal false positivity, which is highly suitable for deployment in high-speed networks.

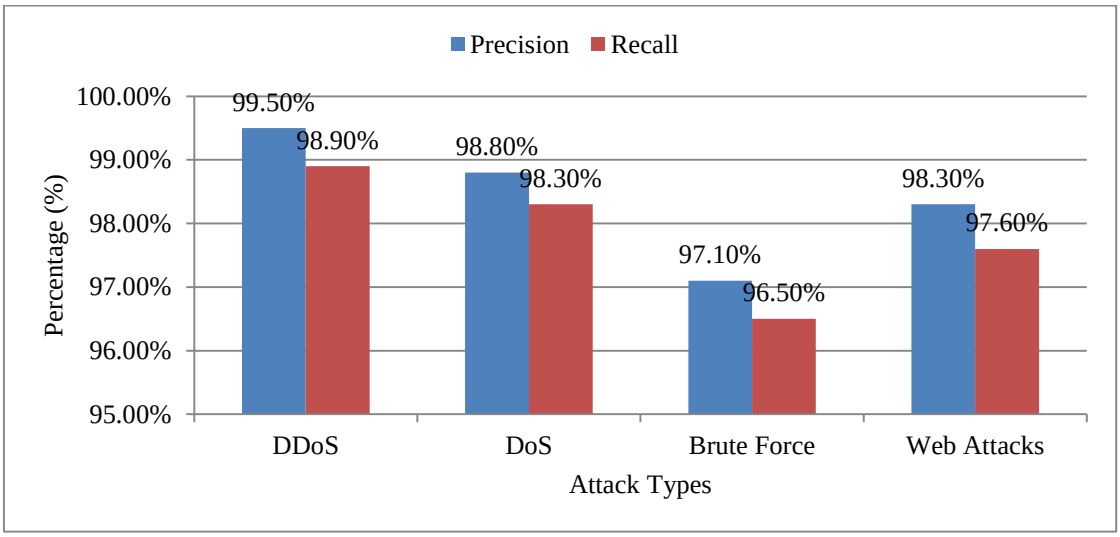

Fig. 17 Precision and recall for attack detection

Figure 17 shows the accuracy and the percentage of recalls obtained by the proposed DPI system, including DDoS, DoS, brute force, and various types of attacks, including web attacks. The system displays excellent detection performance in all classes, with an exact value ranging from 97.1% (brute force) to 99.5% (DDoS) and misses the values between 96.5% (brute force) and 98.9% (DDoS). These results confirm the ability to correctly identify both the DPI engine and the high-volume and secret attacks. The chart recognises the effectiveness of the system in maintaining coherent class-wise detection accuracy for real-time network safety.

Table 9. Processing time comparison

Reference	Delay/Latency
Janabi et al. (2022)	+0.7% delay increase in WAN
Fausto et al. (2022)	<10ms delay (P1); <3ms (P2/P3)
Bocu et al. (2022)	200 ms detection time
Proposed DFA-DPI	0.28 ms latency

Table 9 compares the delay and latency performance of the proposed DFA-DPI system against current solutions. Janabi et al. (2022) registered an increase in delay by 0.7% in WAN environments owing to simulation overhead within Mininet, whereas Fausto et al. (2022) provided delays of below 10 ms for P1 and below 3 ms for P2/P3 with hardware-accelerated switches and DPDK.

Conversely, Bocu et al. (2022) captured a much larger 200 ms detection time; hence, it is not ideal for real-time traffic monitoring. However, the parallelised DFA-DPI system presented in this paper exhibits top-tier performance with an average latency of 0.28 ms, compared to all cited approaches.

This low latency is due to its multi-level parallelism, DFA state minimisation, and effective integration into the SDN environment, allowing high-speed, real-time packet inspection and attack mitigation without causing perceptible delays.

Table. 10 Performance comparison

Method	Method	Accuracy
Janabi et al. (2022)	Naïve Bayes (2022)	98.46%
Fausto et al.(2022)	SM-SDS (2022)	80.3%
Tang et al.(2020)	DeepIDS (2020)	80.7%
Bour et al. (2022)	CNN-based IDS (2022)	94.14%
Proposed Model	DFA-DPI	99.68%

Table 10 and Figure 18 show the accuracy performance of the suggested parallelised DFA-DPI system compared to current intrusion detection mechanisms. The Naïve Bayes model obtained an accuracy rate of 98.46%, whereas SM-SDS and DeepIDS attained lower accuracy rates of 80.3% and 80.7%, respectively, reflecting confined capability for complex attack patterns. The CNN-based IDS showed enhanced performance with a 94.14% accuracy, but it still cannot reach the near-perfect detection rates needed for real-time SDN settings. By contrast, the proposed model surpasses all of these schemes by attaining 99.68% accuracy, which demonstrates its better capability to detect malicious traffic with very few false positives. This notable enhancement results from the optimal DFA state machine, multi-pattern match feature, and parallelised processing scheme that provide high detection accuracy in high-speed networks.

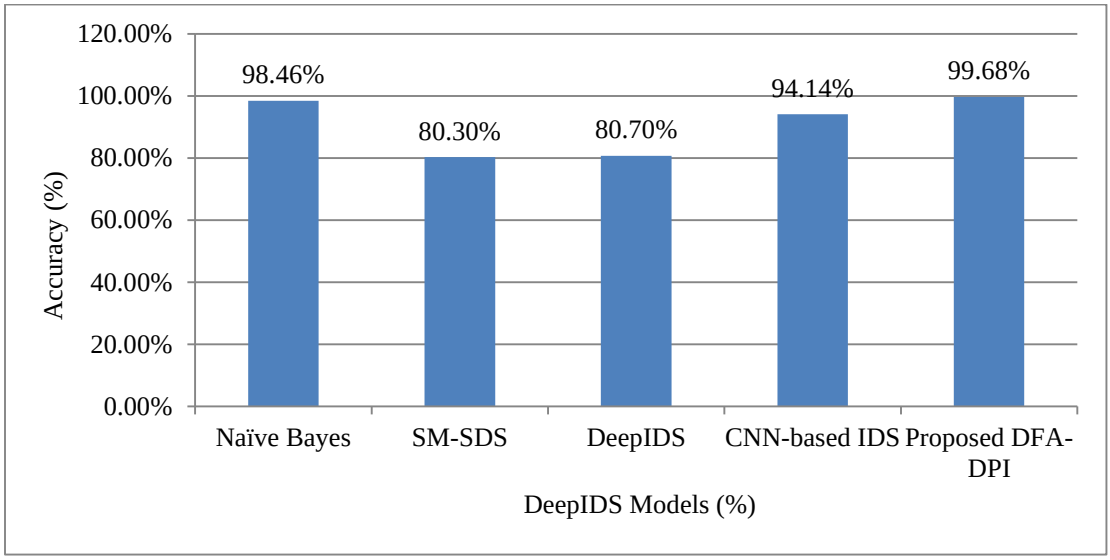

Fig. 18 Accuracy comparison of IDS model

4.2. Discussion

The study introduces a parallel DFA approach using a novel high-performance Deep Packet Inspection (DPI) structure, which is basically unified within a Software-Defined Networking (SDN) environment for detecting and preventing real-time infiltration. The major innovation lies in the capacity of the multi-pattern payload system at high speed, which has taken advantage of advanced parallelisation techniques and customised DFA state machines. Unlike traditional serial DFA-based DPI systems, which struggle

with state explosions and throughput bottlenecks, the proposed model uses Hopcroft's DFA minimisation and CUDA-based GPU acceleration for the matching of parallel patterns. It ensures highly scalable inspection capabilities suitable for a gigabit-scale traffic environment without significant delay. Research focuses on three main innovations: (1) DFA for construction of sequential bite patterns for efficient manufacture using an N-gram encoding to process the payload token, (2) multi-level equality (thread-level, data-level, and pipeline parallelism) to process large traffic volumes in real time, and (3) tight integration. To give. By adopting this approach, the system gained an extraordinary identity accuracy of 99.68%, an accuracy of 99.72% and a minimum average delay of 0.28 ms, improving the current ML-based IDS model and traditional DPI framework. In practical real-time applications, this DPI system operates in the data plane of an SDN-enabled network, which continuously monitors the packets for malicious patterns. On detection of a suspected payload, it communicates with the SDN controller (e.g., Ryu) to update the OpenFlow rules, which can limit instantaneous traffic mitigation functions such as blocking, reunion, or rate. This ensures that the networks remain flexible for high-velocity attacks such as DDoS, brute force efforts, and protocol exploitation. The system has been tested using the CIC-DS 2018 dataset, which provides a realistic attack landscape, validating its appropriateness in the environment of enterprise networks, data centres, and significant infrastructure. Overall, the proposed parallel DFA-DPI system bridges the gap between high-speed traffic inspection and flexible SDN control, making it a strong solution for the modern network, demanding real-time security without renouncing performance. Its highly scalable architecture ensures viability in both hardware-edge devices and large-scale cloud networks.

5. Conclusion

The research proposed a high-performance Deep Packet Inspection (DPI) system, which takes advantage of a parallel DFA approach, adapted to detect and prevent real-time infiltration in the Software-Defined Networking (SDN) environment. By incorporating advanced DFA state minimisation and multi-level equality techniques, the system demonstrated better accuracy (99.68%) and exceptionally low delay (0.28 ms), which enabled efficient packet inspection and dynamic flow mitigation at gigabit traffic rates. Compared to traditional serial DFA and machine learning-based IDS frameworks, the proposed model was found to have terms of its ability to detect, scalability, and accountability, which made it well-suited to deployment in enterprise networks, data centres, and significant infrastructure. SDN ensures flexible and adaptive defence mechanisms against developing the pattern of integration attacks with controllers. For future work, the system can be extended to support encrypted traffic inspection using homomorphic encryption techniques, which enables DPI capabilities without violating privacy obstacles. Additionally, integrating the model detecting AI-powered discrepancies with the DFA engine can increase zero-day attacks and unknown traffic patterns. On hardware-charged platforms such as SmartNICs or FPGAs, patterns can improve throughputs for ultra-high-speed networks (e.g., 100 GBPS). Finally, discovering the DPI architecture distributed in the multi-controller SDN environment will improve mistake tolerance and scalability for large-scale, real-world applications.

References

[1] Amir Ali, and Muhammad Murtaza Yousaf, "Novel Three-Tier Intrusion Detection and Prevention System in Software Defined Network," *IEEE Access*, vol. 8, pp. 109662-109676, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[2] Tamara AlMasri, Mohammad Abu Snober, and Qasem Abu Al-Haija, "IDPS-SDN-ML: An Intrusion Detection and Prevention System Using Software-Defined Networks and Machine Learning," *2022 1st International Conference on Smart Technology, Applied Informatics, and Engineering (APICS)*, Surakarta, Indonesia, pp. 133-137, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Hani Alshahrani et al., "Intrusion Detection Framework for Industrial Internet of Things Using Software Defined Network," *Sustainability*, vol. 15, no. 11, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] Marek Amanowicz, and Damian Jankowski, "Detection and Classification of Malicious Flows in Software-Defined Networks Using Data Mining Techniques," *Sensors*, vol. 21, no. 9, pp. 1-24, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] Celyn Birkinshaw, Elpida Rouka, and Vassilios G. Vassilakis, "Implementing an Intrusion Detection and Prevention System Using Software-Defined Networking: Defending Against Port-Scanning and Denial-of-Service Attacks," *Journal of Network and Computer Applications*, vol. 136, pp. 71-85, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] Razvan Bocu, and Maksim Iavich, "Real-Time Intrusion Detection and Prevention System for 5G and beyond Software-Defined Networks," *Symmetry*, vol. 15, no. 1, pp. 1-15, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Hamideh Bour et al., "A Multi-Layered Intrusion Detection System for Software Defined Networking," *Computers and Electrical Engineering*, vol. 101, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Jonathon Brugman et al., "Cloud Based Intrusion Detection and Prevention System for Industrial Control Systems Using Software Defined Networking," *2019 Resilience Week (RWS)*, San Antonio, TX, USA, pp. 98-104, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[9] Sotiris Chatzimiltis et al., "A Collaborative Software Defined Network-Based Smart Grid Intrusion Detection System," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 700-711, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[10] Qiumei Cheng et al., "Machine Learning Based Malicious Payload Identification in Software-Defined Networking," *Journal of Network and Computer Applications*, vol. 192, pp. 1-12, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[11] Xavier Etxezarreta Argarate, "*Software-Defined Networking Approaches for Intrusion Response in Industrial Control Systems*," Thesis, Mondragon University, pp. 1-193, 2024. [Google Scholar] [Publisher Link]

[12] Alessandro Fausto et al., "Reduction of the Delays within an Intrusion Detection System (IDS) Based on Software Defined Networking (SDN)," *IEEE Access*, vol. 10, pp. 109850-109862, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Jalal Ghadermazi, Ankit Shah, and Nathaniel D. Bastian, "Towards Real-Time Network Intrusion Detection with Image-Based Sequential Packets Representation," *IEEE Transactions on Big Data*, vol. 11, no. 1, pp. 157-173, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[14] Guo Guangfeng, Zhang Junxing, and Ma Zhanfei, "Intrusion Prevention with Attack Traceback and Software-Defined Control Plane for Campus Networks," *Computer Science and Information Systems*, vol. 18, no. 3, pp. 867-891, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[15] Abdinasir Hirsi et al., "Detecting DDoS Threats Using Supervised Machine Learning for Traffic Classification in Software Defined Networking," *IEEE Access*, vol. 12, pp. 166675-166702, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[16] Ahmed H. Janabi, Triantafyllos Kanakis, and Mark Johnson, "Overhead Reduction Technique for Software-Defined Network Based Intrusion Detection Systems," *IEEE Access*, vol. 10, pp. 66481-66491, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[17] Ahmed H. Janabi, Triantafyllos Kanakis, and Mark Johnson, "Survey: Intrusion Detection System in Software-Defined Networking," *IEEE Access*, vol. 12, pp. 164097-164120, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[18] Keagan Jarvis, "*Network Intrusion Prevention in the Evolved Packet Core Utilising Software Defined Networks and Network Function Virtualisation*," Master Thesis, 2019. [Google Scholar] [Publisher Link]

[19] M. Kokila M Kokila, and Srinivasa Srinivasa Reddy Konda, "DeepSDN: Deep Learning Based Software Defined Network Model for Cyberthreat Detection in IoT Network," *ACM Transactions on Internet Technology*, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[20] Auther Makuvaza, Dharm Singh Jat, and Attlee M. Gamundani, "Deep Neural Network (DNN) Solution for Real-time Detection of Distributed Denial of Service (DDoS) Attacks in Software Defined Networks (SDNs)," *SN Computer Science*, vol. 2, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[21] Hanan Mustapha et al., "Rethinking Deep Packet Inspection Design and Deployment in the era of SDN and NFV," *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, Haikou, Hainan, China, pp. 1505-1514, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[22] Talha Naqash, Sajjad Hussain Shah, and Muhammad Najam Ul Islam, "Statistical Analysis Based Intrusion Detection System for Ultra-High-Speed Software Defined Network," *International Journal of Parallel Programming*, vol. 50, pp. 89-114, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[23] Edeh Michael Onyema et al., "A Security Policy Protocol for Detection and Prevention of Internet Control Message Protocol Attacks in Software Defined Networks," *Sustainability*, vol. 14, no. 19, pp. 1-19, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[24] Kunkun Rui, Hongzhi Pan, and Sheng Shu, "Secure Routing in the Internet of Things (IoT) with Intrusion Detection Capability Based on Software-Defined Networking (SDN) and Machine Learning Techniques," *Scientific Reports*, vol. 13, no. 1, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[25] Ayodeji Olalekan Salau, and Melesew Mossie Beyene, "Software Defined Networking Based Network Traffic Classification Using Machine Learning Techniques," *Scientific Reports*, vol. 14, no. 1, pp. 1-16, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[26] N. Satheesh et al., "Flow-Based Anomaly Intrusion Detection Using Machine Learning Model with Software Defined Networking for OpenFlow Network," *Microprocessors and Microsystems*, vol. 79, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[27] Wenguang Song et al., "A Software Deep Packet Inspection System for Network Traffic Analysis and Anomaly Detection," *Sensors*, vol. 20, no. 6, pp. 1-14, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[28] Tuan Anh Tang et al., "DeepIDS: Deep Learning Approach for Intrusion Detection in Software Defined Networking," *Electronics*, vol. 9, no. 9, pp. 1-18, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[29] Sultan Zavrak, and Murat Iskefiyeli, "Flow-Based Intrusion Detection on Software-Defined Networks: A Multivariate Time Series Anomaly Detection Approach," *Neural Computing and Applications*, vol. 35, no. 16, pp. 12175-12193, 2023. [CrossRef] [Google Scholar] [Publisher Link]