Advancing Deep Packet Inspection in SDNs: P4 vs OpenFlow Across Control and Data Planes

Anthony J. Bustamante, Marc J. Dupuis, Brent Lagesse
University of Washington Bothell
Email: absuarez@uw.edu, marcjd@uw.edu, lagesse@uw.edu

***Abstract*—This paper examines Deep Packet Inspection (DPI) capabilities within Software-Defined Networking (SDN), comparing the efficacy of P4 programming language with the traditional OpenFlow protocol. While OpenFlow is fundamental to SDN, it is limited in DPI due to its focus on lower OSI layers and its limited configurability. In contrast, P4 offers enhanced flexibility, enabling us to define operations at the application layer beyond what OpenFlow offers. Through emulations using Open vSwitch and BMv2 switches, this study compares Openflow and P4 for DPI, particularly in handling HTTP and SQL protocols. The research provides three distinct DPI implementations, benchmarking their advantages and disadvantages. The findings in terms of accuracy, latency, resource consumption, and throughput validate P4's superiority in managing DPI tasks, contributing to advancements in SDN functionalities and paving the way for future innovations in network security and optimization.**

***Index Terms*—Openflow, P4, Data Plane Programmability, Software Defined Networking, HTTP, SQL, Web filtering, Command Injection, gRPC.**

I. Introduction

Software-Defined Networking (SDN) has revolutionized network management by introducing programmability and centralized control, allowing for dynamic adjustments to network behavior [1]. SDN separates the control and data planes, enabling network configurations without the limitations of hardware. This study focuses on enhancing Deep Packet Inspection (DPI) within SDN, particularly at the application layer, a domain traditionally managed by protocols like OpenFlow [2].

As cloud computing and other paradigms evolve, network security threats are becoming more sophisticated [3], necessitating advanced DPI methods capable of inspecting deeper into packet payloads [4]. Traditional protocols like OpenFlow are limited in their ability to handle application-layer traffic, highlighting the need for more flexible solutions.

This research proposes the P4 programming language as a solution to overcome the limitations of OpenFlow. P4 enables more granular control in the data plane, allowing for detailed inspection of network traffic beyond the capabilities of traditional methods. By leveraging P4's flexibility, this study aims to validate its superiority over OpenFlow in executing DPI tasks.

The experimental setup involves using OVS and BMv2 switches and advanced protocols like gRPC and P4Runtime to create a robust DPI framework within SDN environments [5]. This study compares different implementations of DPI, evaluating their effectiveness in managing network threats and their impact on performance metrics such as latency, accuracy, and resource usage.

By providing a comprehensive analysis of P4 and OpenFlow in DPI contexts, this research contributes to advancing SDN functionalities, particularly in enhancing network security and management. The findings aim to pave the way for future innovations in increasingly complex network environments.

II. Related Work

The integration of Deep Packet Inspection (DPI) within Software-Defined Networking (SDN) has been explored through various approaches in existing literature, emphasizing the transformative potential of SDN and DPI.

Gupta et al. [5] introduce a DPI implementation using packet recirculation within P4 networks, demonstrating the feasibility of deep packet inspection without significant performance penalties. This work focuses on P4's capabilities but does not compare it comprehensively with OpenFlow, which is the aim of this paper.

Ahad et al. [6] present DPI-based Intrusion Prevention Systems (IPS) for P4 programmable networks, specifically for DNS traffic. However, their work lacks a comparative analysis with OpenFlow, which this paper addresses by exploring DPI across multiple applications like HTTP and SQL.

Hypolite et al. [7] propose DeepMatch, a system for implementing DPI within the data plane using network processors. While this research showcases technical potential, it does not integrate broader SDN frameworks or evaluate comprehensive performance metrics, which are key aspects of this paper.

Li et al. [8] explore DPI for application-aware traffic control in SDNs, enhancing Quality of Service (QoS) through dynamic traffic management. Their approach aligns with this paper's objectives of improving network management via DPI, but this work extends the analysis to a broader integration with SDN technologies, including performance evaluation.

Chin et al. [9] investigate selective packet inspection for mitigating DoS attacks using SDN's centralized control. While effective for specific threats, their focus is narrower compared to this paper, which aims to employ DPI for a wider range of attack vectors and integrate it within a comprehensive SDN framework using P4 and OpenFlow.

This paper builds upon these contributions by proposing an integrative framework that leverages SDN technologies for

comprehensive DPI implementation, aiming to enhance network security, management, and performance across diverse environments.

III. Objectives and Research Hypotheses

This research aims to achieve the following specific objectives:

- Investigate the flexibility and programmability of P4 in implementing DPI, especially at the application layer (Layer 7), and compare its performance with OpenFlow.
- Develop and test multiple DPI implementations using both P4 and OpenFlow within an SDN environment to evaluate their respective strengths and limitations.
- Benchmark the performance of P4-based DPI implementations against OpenFlow-based solutions in terms of accuracy, latency, and resource utilization.
- Provide a comprehensive analysis that not only validates P4's potential advantages over OpenFlow but also contributes to advancing SDN technologies for enhanced network security and management.

A. Research Hypotheses

This paper is based on the following hypotheses:

1) ***Hypothesis One: Flexibility and Application-layer Analysis****:* The flexibility offered by the P4 programming language will allow for more effective Deep Packet Inspection (DPI) at the application layer, compared to the traditional OpenFlow protocol(Control plane approach). This flexibility enables better adaptability to emerging network threats and a more comprehensive analysis of application-layer traffic.

2) ***Hypothesis Two: Performance and Efficiency****:* Implementing DPI using P4 will result in faster and more efficient network security measures by leveraging data plane programmability. This is expected to lead to quicker detection and response to potential threats, with reduced latency and improved overall network performance.

3) ***Hypothesis Three: Resource Usage****:* DPI implemented with P4 will consume fewer computing resources than DPI approaches that rely heavily on the control plane, as seen with OpenFlow-based methods. This hypothesis suggests that a data plane-centric DPI strategy can provide a more resource-efficient solution for managing network security.

These hypotheses will be tested through rigorous experimentation and analysis, comparing the performance of P4 and OpenFlow in handling various network security challenges. Key metrics such as processing speed, resource consumption, throughput, and the accuracy of DPI will be used to validate or refute these hypotheses.

IV. Development of Components

A. DPI Implementation via OpenFlow and Open vSwitch

The first configuration employs Mininet to create a basic network topology using OpenFlow and Open vSwitch (see Fig. 1). The setup includes a switch connected to client and server hosts, with the switch forwarding all traffic to a remote POX controller for processing. DPI is conducted at the controller level due to the static nature of OpenFlow's data plane, which limits the inspection of dynamic-sized headers such as those in HTTP or SQL traffic. The POX controller runs a custom DPI algorithm that decodes TCP payloads, inspects HTTP and SQL traffic, and applies predefined actions based on the analysis.

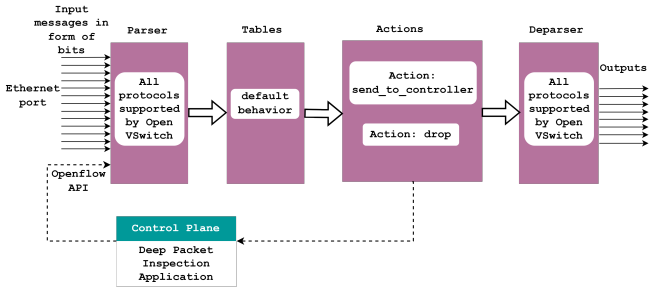

Fig. 1. Traffic flow representation for the Openflow implementation

B. Integration of BMv2 and P4runtime for DPI

In Figure 2 we can see the second configuration integrates the BMv2 switch, a P4-programmable environment, with Mininet. This setup enables more granular control over the data plane, facilitating DPI tasks directly through P4Runtime. A custom Mininet class was developed to support BMv2, allowing for the deployment of a P4-programmable switch. The BMv2 switch is initialized with P4Runtime support and a DPI-focused P4 program, which handles packet parsing, header inspection, and traffic management. The controller uses P4Runtime to dynamically inject P4 rules and manage network flows based on real-time DPI analysis.

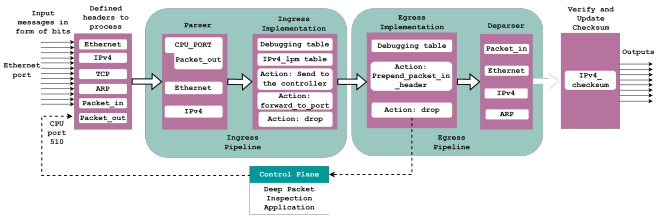

Fig. 2. Traffic flow representation for the P4Runtime implementation

C. Data Plane-Centric DPI Strategy

The third configuration shifts DPI processing entirely to the data plane, utilizing P4 programming to parse and inspect packets directly within the network device (see Fig. 3). This method would minimize latency by eliminating the need for control plane intervention. The data plane-centric approach involves defining standard and custom headers (e.g., HTTP, SQL), implementing a parser to extract these headers, and applying DPI logic directly within the ingress pipeline. The egress pipeline ensures efficient packet forwarding post-inspection. This configuration showcases the potential of P4 in executing sophisticated DPI tasks within the data plane, reducing latency while maintaining high inspection accuracy.

For this implementation we did not use a controller, instead, we introduced flow table rules with APIs like thrift.

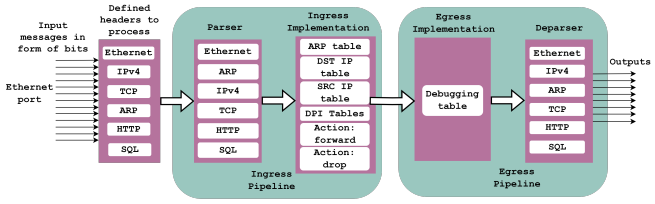

Fig. 3. Traffic flow representation for the P4 Data plane implementation

D. Operational Methodology and Performance Considerations

Each configuration is tested for its DPI capabilities using a variety of metrics, including latency, accuracy, resource consumption, and throughput. The performance of these DPI systems is critically evaluated to understand their strengths and limitations. The data plane-centric approach, while promising in reducing latency, poses challenges in programming complexity and flexibility compared to the control plane-based solutions. These configurations collectively contribute to advancing the understanding of DPI in SDN environments, highlighting the trade-offs between control plane and data plane processing.

For the performance computation, we calculated the percentage change using a formula that compares the metric value of the evaluated DPI implementation (P4 Data Plane or P4runtime) to that of the baseline DPI implementation (OpenFlow-based). Overall, this section serves as the foundation for subsequent experimentation and analysis, aiming to validate the research hypotheses concerning the efficacy of P4 versus OpenFlow for DPI tasks within SDN frameworks.

V. Tests and Results Summary

This section evaluates the Deep Packet Inspection (DPI) techniques implemented for enhancing network security within Software-Defined Networking (SDN) environments, focusing on HTTP URL filtering and SQL command filtering. Each DPI implementation—OpenFlow-based, P4Runtime-based, and P4 Data Plane-Centric—is tested and compared using metrics such as accuracy, delay, resource utilization, and throughput.

A. HTTP URL Filtering

HTTP URL filtering inspects HTTP GET requests against a blacklist to block unauthorized web access. The DPI implementations include OpenFlow-based, P4Runtime-based, and P4 Data Plane-Centric approaches.

1) Algorithmic Implementation: The core logic, shown in Figure 4, processes incoming HTTP packets to determine whether they should be dropped based on blacklisted hosts. The OpenFlow and P4Runtime implementations use this algorithm, while the P4 Data Plane version executes it directly in the data plane, enabling faster filtering. Relevant flow rules using hexadecimal encoding for domain recognition are listed in Table I.

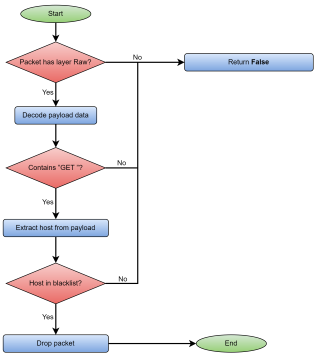

Fig. 4. URL filtering application algorithm

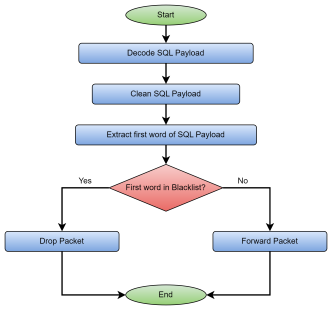

Fig. 5. SQL command filtering algorithm

2) Performance Metrics: Performance was measured using the following metrics:

- **Accuracy:** All implementations achieved high accuracy, with minimal variation in metrics such as precision, recall, and F1 score, as shown in Table II.
- **Delay:** P4 Data Plane-Centric DPI had the lowest delay, demonstrating superior responsiveness compared to other implementations (Table III).
- **Resource Utilization:** P4 Data Plane-Centric DPI was the most efficient, with lower execution time, CPU usage, and memory consumption (Table IV).
- **Throughput:** The P4 Data Plane implementation showed higher throughput, particularly with larger packet sizes, as seen in Figure 6, indicating its suitability for high-throughput scenarios.

TABLE II
ACCURACY METRICS FOR HTTP URL FILTERING AND SQL COMMAND FILTERING ACROSS DIFFERENT DPI IMPLEMENTATIONS.

Implementation	Accuracy	Precision	Recall	F1 Score	TPR	FPR
HTTP OpenFlow-based DPI	0.99	1.00	0.99	0.99	1.00	0.01
HTTP P4runtime-based DPI	0.99	1.00	0.98	0.99	1.00	0.02
HTTP Data Plane-Centric DPI	0.99	1.00	0.98	0.99	1.00	0.02
SQL OpenFlow-based DPI	0.6061	-	-	-	-	-
SQL P4runtime-based DPI	0.5152	-	-	-	-	-
SQL Data Plane-Centric DPI	0.7576	-	-	-	-	-

B. SQL Command Filtering

SQL command filtering inspects traffic directed to SQL servers, blocking unauthorized commands. The evaluation covers OpenFlow-based, P4Runtime-based, and P4 Data Plane-Centric DPI implementations.

1) Algorithmic Implementation: Filtering logic (Figure 5) intercepts and processes SQL commands, taking action based on predefined rules. The P4 Data Plane version processes these commands directly in the data plane, utilizing entries in the application flow table to filter based on hexadecimal signatures (Table I).

2) Performance Metrics: The performance metrics include:

- **Delay:** P4 Data Plane-Centric DPI again demonstrated the lowest delay, confirming its efficiency (Table III).

TABLE I
SOME ENTRIES IN THE FILTER METHOD AND COMMAND TABLE FOR HTTP AND SQL BASED ON DOMAIN BLACKLISTING AND ENCODED SIGNATURES.

Entry No.	Method Match/Command Signature	Host/Hexadecimal Match	Wildcard	Action
HTTP Entry 1	GET(0x47455420)	0x00000000000064306471793177653071336 62e636f6d0000000000000000	0x00000000000ffffffffffffffffffffffffffffffffffff0000000000000000	drop
HTTP Entry 5	GET(0x47455420)	0x00000000000072623374677474322e636f6d0000000000000000000000000	0x00000000000ffffffffffffffffffffffffffffffffffff0000000000000000000000	drop
SQL Entry 1	DROP	0x44524F5000000000000000000000000	0xffffffff0000000000000000000000000	drop
SQL Entry 2	SHOW	0x53484F5700000000000000000000000	0xffffffff0000000000000000000000000	drop
SQL Entry 3	GRANT	0x4752414E5400000000000000000000000	0xffffffffffff0000000000000000000000	drop
SQL Entry 4	REVOKE	0x5245564F4B4500000000000000000000	0xffffffffffffff0000000000000000000	drop
SQL Entry 5	CREATE	0x4352454154450000000000000000000	0xffffffffffffff0000000000000000000	drop
SQL Entry 6	USE	0x5553450000000000000000000000000	0xffffff00000000000000000000000000	drop

TABLE III
RESPONSE TIME METRICS FOR HTTP URL FILTERING AND SQL COMMAND FILTERING APPLICATIONS ACROSS DIFFERENT DPI IMPLEMENTATIONS.

Implementation	HTTP Min Delay (ms)	HTTP Avg Delay (ms)	HTTP Max Delay (ms)	SQL Min Delay (ms)	SQL Avg Delay (ms)	SQL Max Delay (ms)
OpenFlow-based DPI	30.06	97.07	196.21	247.69	391.45	1139.55
P4runtime-based DPI	97.01	236.12	1224.57	241.32	779.43	2054.51
Data Plane-Centric DPI	6.25	33.23	156.07	42.01	346.40	1853.58

TABLE IV
PERFORMANCE METRICS FOR HTTP AND SQL DPI IMPLEMENTATIONS

Implementation	Execution Time (ms)	Peak CPU Usage (%)	CPU Usage Increment Factor	CPU Time (s)	Max Memory Usage (KB)
P4runtime-based DPI (HTTP)	491242	57%	57.00	2.44s	99072
P4runtime-based DPI (SQL)	63535	65.44%	40.90	2.38s	92368
OpenFlow-based DPI (HTTP)	464454	43.03%	43.03	2.42s	98876
OpenFlow-based DPI (SQL)	46544	65.2%	65.20	2.31s	98272
Data Plane-Centric DPI (HTTP)	286865	34.62%	33.61	1.79s	98860
Data Plane-Centric DPI (SQL)	54430	63.3%	63.30	2.43s	92488

- **Accuracy:** This implementation also achieved the highest accuracy for SQL filtering (Table II).
- **Resource Utilization:** Resource usage metrics (Table IV) indicate balanced and efficient utilization in the P4 Data Plane approach.
- **Throughput:** The P4 Data Plane-Centric DPI consistently outperformed the others across packet sizes, supporting its effectiveness in high-throughput conditions (Figure 8).

A summary of the results for our Web and SQL Command Filtering Applications can be seen in the figures 10 and 12.

VI. INTERPRETATION OF RESULTS

The table VI presents the percentage change in performance metrics of the P4runtime-based and Data Plane-Centric DPI implementations relative to the OpenFlow-based DPI implementation for both HTTP and SQL applications.

1) Accuracy: P4runtime-based DPI (HTTP) shows no percentage change in accuracy compared to OpenFlow-based DPI, maintaining a stable performance. However, P4runtime-based DPI (SQL) shows a 15% reduction in accuracy, suggesting that for SQL command filtering, P4runtime-based DPI might struggle to maintain the same accuracy levels as OpenFlow. On the other hand, Data Plane-Centric DPI (HTTP) similarly shows no difference in accuracy compared to OpenFlow-based DPI, while Data Plane-Centric DPI (SQL) shows a notable improvement of 25%, indicating that this approach is significantly more accurate for SQL command filtering compared to OpenFlow.

2) Average Delay/Time Response: Figures 7 and 9, along with heatmaps in Figures 11 and 13, illustrate the delay differences across P4runtime-based, Data Plane-Centric, and OpenFlow-based DPI implementations for HTTP and SQL traffic. For HTTP traffic, P4runtime-based DPI shows significantly higher average delay (+143.34%) compared to OpenFlow-based DPI, as reflected by the frequent delay spikes in Figure 7. The heatmap (Figure 11) further confirms this with red areas indicating greater delays. In contrast, Data Plane-Centric DPI reduces average delay by 65.77%, as shown by the consistently lower delay in Figure 7 and the blue regions in Figure 11, which indicate lower delays compared to OpenFlow-based DPI. Similarly, for SQL traffic, P4runtime-based DPI increases delay by 99.13%, with peaks exceeding 2000 ms in Figure 9. The heatmap in Figure 13 again highlights these delay increases. However, Data Plane-Centric DPI achieves a modest reduction in delay (-11.51%), as indicated by the more stable delay line in Figure 9 and the blue areas in Figure 13.

In conclusion, Data Plane-Centric DPI consistently outperforms P4runtime-based DPI and OpenFlow-based DPI in terms of lower delays, especially for HTTP traffic, making it a better choice for low-latency environments.

3) Execution Time: The execution time for P4runtime-based DPI (HTTP) increases by 5.76%, indicating a slight performance overhead compared to OpenFlow. For SQL traffic, this increase is more pronounced, with a 36.51% rise in execution time, showing that P4runtime-based DPI is less efficient in processing SQL traffic. Data Plane-Centric DPI (HTTP), on the other hand, shows a significant decrease in execution time (-38.25%) compared to OpenFlow, making it much faster for HTTP processing. For SQL traffic, Data Plane-Centric DPI (SQL) still shows an increase in execution time (+16.95%), but this increase is much less than that seen in P4runtime-based DPI.

4) Peak CPU Usage: The P4runtime-based DPI (HTTP) has a much higher peak CPU usage (+32.43%) compared to

TABLE VI
PERCENTAGE CHANGE IN PERFORMANCE METRICS COMPARED TO OPENFLOW-BASED DPI FOR HTTP AND SQL APPLICATIONS

Implementation	Accuracy (%)	Avg Delay (%)	Execution Time (%)	Peak CPU Usage (%)	CPU Time (%)	Max Memory Usage (%)
P4runtime-based DPI (HTTP)	0.00%	+143.34%	+5.76%	+32.43%	+0.83%	+0.20%
P4runtime-based DPI (SQL)	-15.00%	+99.13%	+36.51%	+0.37%	+3.03%	-6.02%
Data Plane-Centric DPI (HTTP)	0.00%	-65.77%	-38.25%	-19.55%	-26.03%	-0.02%
Data Plane-Centric DPI (SQL)	+25.00%	-11.51%	+16.95%	-2.91%	+5.19%	-5.89%

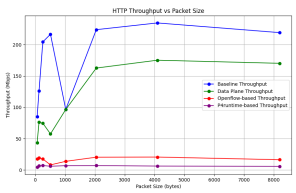

Fig. 6. Throughput for Web Filtering Application

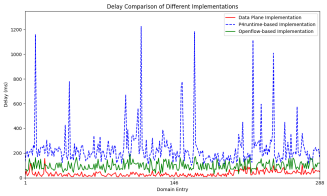

Fig. 7. Delay for URL Filtering Application

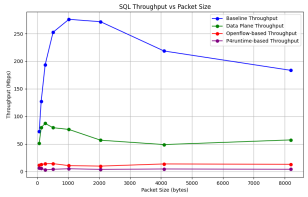

Fig. 8. Throughput for SQL Command Filtering Application

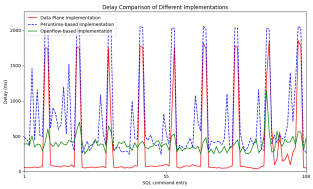

Fig. 9. Delay comparison for SQL Command Filtering Application

OpenFlow-based DPI, indicating higher resource consumption. This pattern is also observed in SQL traffic, where P4runtime-based DPI (SQL) shows a marginal increase (+0.37%) in CPU usage. In contrast, Data Plane-Centric DPI (HTTP) shows a decrease in peak CPU usage (-19.55%) compared to OpenFlow, demonstrating more efficient resource usage for HTTP traffic. Similarly, Data Plane-Centric DPI (SQL) shows a reduction in peak CPU usage (-2.91%), but the reduction is less dramatic than for HTTP traffic.

5) CPU Time: For both HTTP and SQL traffic, P4runtime-based DPI (HTTP) and P4runtime-based DPI (SQL) show slight increases in CPU time (+0.83% and +3.03%, respectively), indicating a minor increase in processing time for these DPI implementations. Data Plane-Centric DPI (HTTP) and Data Plane-Centric DPI (SQL) show reductions in CPU time (-26.03% and -5.19%, respectively), highlighting that this approach is generally more efficient in terms of processing time for both HTTP and SQL traffic.

6) Max Memory Usage: Finally, P4runtime-based DPI (HTTP) shows only a small increase in memory usage (+0.20%) compared to OpenFlow-based DPI, whereas P4runtime-based DPI (SQL) exhibits a reduction in memory usage (-6.02%). Similarly, Data Plane-Centric DPI (HTTP) demonstrates a minimal reduction in memory usage (-0.02%), while Data Plane-Centric DPI (SQL) shows a more substantial reduction (-5.89%), indicating that both implementations consume less memory for SQL processing compared to OpenFlow.

7) Throughput Results: Figures 6 and 8 and Table VII show the throughput results for HTTP and SQL traffic across different DPI implementations as the packet size increases. For both HTTP and SQL, the Baseline Throughput is significantly higher than the other implementations. As expected, the Data Plane-Centric DPI consistently provides better throughput than the OpenFlow-based and P4runtime-based implementations. Specifically, the throughput for P4runtime-based DPI is the lowest in both cases, indicating that it has the highest overhead for traffic processing. For HTTP traffic, the Data Plane-Centric DPI performs comparably to the Baseline throughput, especially for packet sizes over 1,000 bytes. However, there is a noticeable drop in performance for the smaller packet sizes. Similarly, for SQL traffic, Data Plane-Centric DPI outperforms the OpenFlow-based DPI and P4runtime-based DPI, although it still falls short compared to the Baseline throughput.

These results indicate that Data Plane-Centric DPI offers a more scalable and efficient solution compared to OpenFlow-based and P4runtime-based DPI, particularly for larger packet sizes.

8) Limitations of the Study: The experiments were conducted in a controlled environment, which may not fully reflect the complexity of real-world network conditions. Additionally, this research focused solely on unencrypted traffic. Despite these limitations, the study's validity is reinforced by testing the implementations in two different runtime environments, which consistently produced results aligned with those presented in this paper. For those interested in replicating or extending this work, the project is accessible on our **Github Repository**.

VII. CONCLUSIONS

This research critically evaluated the capabilities of Deep Packet Inspection (DPI) within Software-Defined Networking (SDN) frameworks, comparing P4 programming with traditional OpenFlow protocol. Through benchmarking and emulation on Open vSwitch and BMv2 switches, we analyzed different DPI implementations on metrics such as accuracy, delay, throughput, execution time, CPU usage, and memory consumption. The results provide clear evidence supporting our hypotheses:

1. **Flexibility and Application-Layer Analysis**: The **Data Plane-Centric DPI** consistently demonstrated either the same or higher accuracy compared to **Control-Plane-based DPI**, particularly achieving 25.00% higher accuracy in SQL command filtering (Openflow Implementation). This confirms that P4's flexibility enables DPI on the data plane without sacrificing quality.

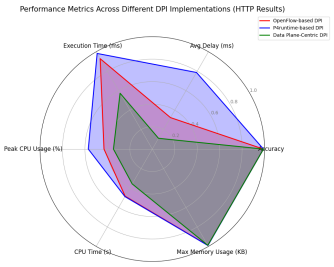

Fig. 10. Scaled values for performance metrics for URL Filtering Application

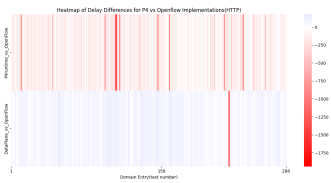

Fig. 11. Heatmap comparison for URL Filtering Application (P4 vs Openflow)

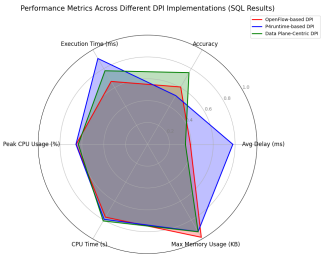

Fig. 12. Scaled values for performance metrics for SQL Command Filtering Application

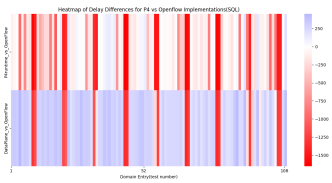

Fig. 13. Heatmap comparison for SQL command filtering application (P4 vs Openflow)

TABLE VII
THROUGHPUT COMPARISON FOR HTTP AND SQL TRAFFIC ACROSS DIFFERENT IMPLEMENTATIONS (MBPS)

Packet Size (Bytes)	Baseline HTTP	Data Plane HTTP	OpenFlow HTTP	P4Runtime HTTP	Baseline SQL	Data Plane SQL	OpenFlow SQL	P4Runtime SQL
64	84.94	43.43	17.87	4.52	72.77	51.42	11.92	6.62
128	126.29	76.17	19.35	6.52	127.42	79.66	12.63	5.86
256	204.54	74.59	17.49	7.03	193.55	87.32	14.19	3.16
512	216.78	57.57	8.11	5.96	252.57	79.53	14.40	4.20
1024	96.71	96.54	13.79	6.83	275.99	76.40	10.77	5.08
2048	224.04	162.75	20.40	6.99	271.65	56.86	9.93	4.00
4096	234.81	175.28	20.61	6.23	218.61	48.94	13.83	4.70
8192	219.36	170.32	16.38	5.73	183.54	57.38	13.17	4.19

2. **Performance and Efficiency**: The **Data Plane-Centric DPI** significantly outperformed the **Control-Plane-based DPI** in terms of latency, with 65.77% faster HTTP filtering and 11.51% faster SQL filtering(Openflow Implementation). This validates the hypothesis that P4 can enhance DPI performance by enabling real-time traffic analysis with minimal delay.

3. **Resource Usage**: The **Data Plane-Centric DPI** consumed fewer resources in terms of CPU time and peak CPU usage compared to **Control-Plane-based DPI**, further validating our hypothesis of P4's efficiency in resource management.

Also, the **P4runtime-based DPI**, which involved a controller, consistently exhibited worse performance compared to **OpenFlow-based DPI**, both in terms of delay and resource usage. This suggests that while P4 offers substantial benefits when focusing on the data plane, its advantages diminish when coupled with a control plane controller.

The percentage change analysis further reinforced these findings, highlighting the consistent performance improvements of the **Data Plane-Centric DPI** in terms of lower average delay, reduced resource consumption, and higher accuracy compared to **OpenFlow-based DPI**. Conversely, the **P4runtime-based DPI** incurred higher delays and resource usage while maintaining good accuracy.

In conclusion, this research confirms that advanced data plane programmability with P4 offers a superior alternative to traditional Control-plane-based DPI (either Openflow or P4runtime based) for real-time traffic analysis and network security. By offloading inspection tasks to the data plane, P4 enables in-depth packet inspection without overburdening the control plane, thus paving the way for future innovations in network management and SDN technologies.

REFERENCES

[1] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," in *Proceedings of the IEEE*, vol. 103, 2015, pp. 14–76.

[2] G. Bianchi, M. Bonola, A. Capone, and C. Cascone, "Openstate: Programming platform-independent stateful openflow applications inside the switch," vol. 44, 04 2014, pp. 44–51.

[3] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: network processing as a cloud service," *SIGCOMM Comput. Commun. Rev.*, vol. 42, pp. 13–24, 2012.

[4] Enea, "The future of deep packet inspection: Key findings from the enea dpi survey," 2020. [Online]. Available: https://www.enea.com/insights/the-future-of-deep-packet-inspection-key-findings-from-the-enea-dpi-survey

[5] S. Gupta, D. Gosain, M. Kwon, and H. B. Acharya, "Deep4r: Deep packet inspection in p4 using packet recirculation," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 2023, pp. 1–10.

[6] A. Ahad, R. A. Bakar, M. Arslan, and M. H. Ali, "Dpidns:a deep packet inspection based ips for security of p4 network data plane," in *2023 International Conference on Smart Computing and Application (ICSCA)*, 2023, pp. 1–8.

[7] J. Hypolite, J. Sonchack, S. Hershkop, N. Dautenhahn, A. DeHon, and J. M. Smith, "Deepmatch: practical deep packet inspection in the data plane using network processors," in *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 336–350. [Online]. Available: https://doi.org/10.1145/3386367.3431290

[8] G. Li, M. Dong, K. Ota, J. Wu, J. Li, and T. Ye, "Deep packet inspection based application-aware traffic control for software defined networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–6.

[9] T. Chin, X. Mountrouidou, X. Li, and K. Xiong, "Selective packet inspection to detect dos flooding using software defined networking (sdn)," in *2015 IEEE 35th International Conference on Distributed Computing Systems Workshops*, 2015, pp. 95–99.