Domain Name Security Inspection at Line Rate: TLS SNI Extraction in the Data Plane Using P4 and DPDK

Ali Mazloum, Ali AlSabeh, Elie Kfoury, Jorge Crichigno
University of South Carolina, USA

GMI-AIMS-5
14 February 2025

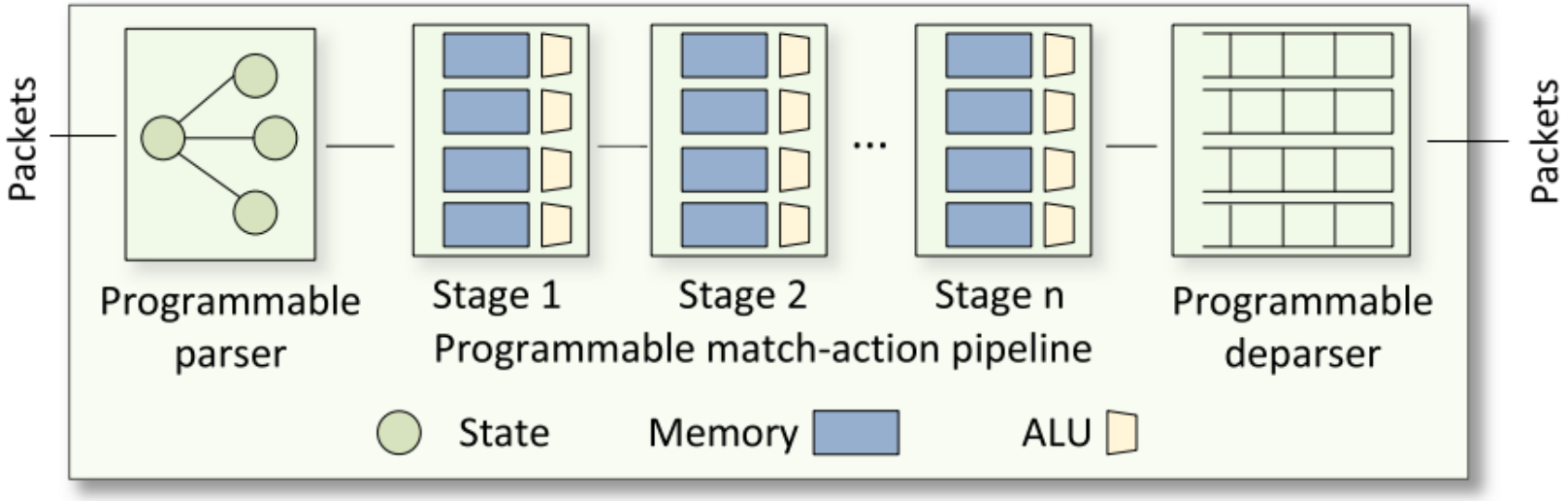
San Diego Supercomputer Center

Background: TLS

- Transport Layer Security (TLS) is a cryptographic protocol that provides end-to-end security for data transmitted over an insecure network
- TLS supports multiple extensions to facilitate end-to-end communication
- One of these extensions is the Server Name Indication (SNI)
- Modern security applications filter and block malicious traffic using SNI

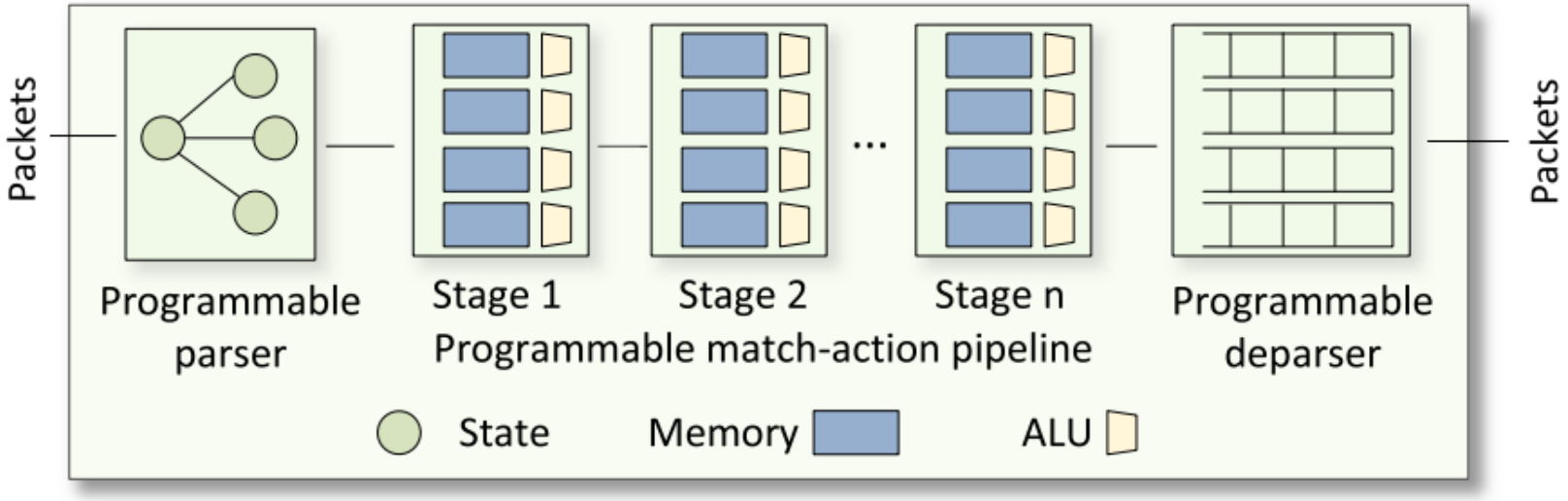
Background: PDP Switches and P4

- Programmable Data Plane (PDP) switches are network devices that allow the user to define the packet processing logic in the data plane
- P4, or Programming Protocol-Independent Packet Processors, is a widely adopted language to program the PDP switches
- The main three components of the P4 PDP switches are the programmable parser, the programmable match-action pipeline, and the programmable deparser



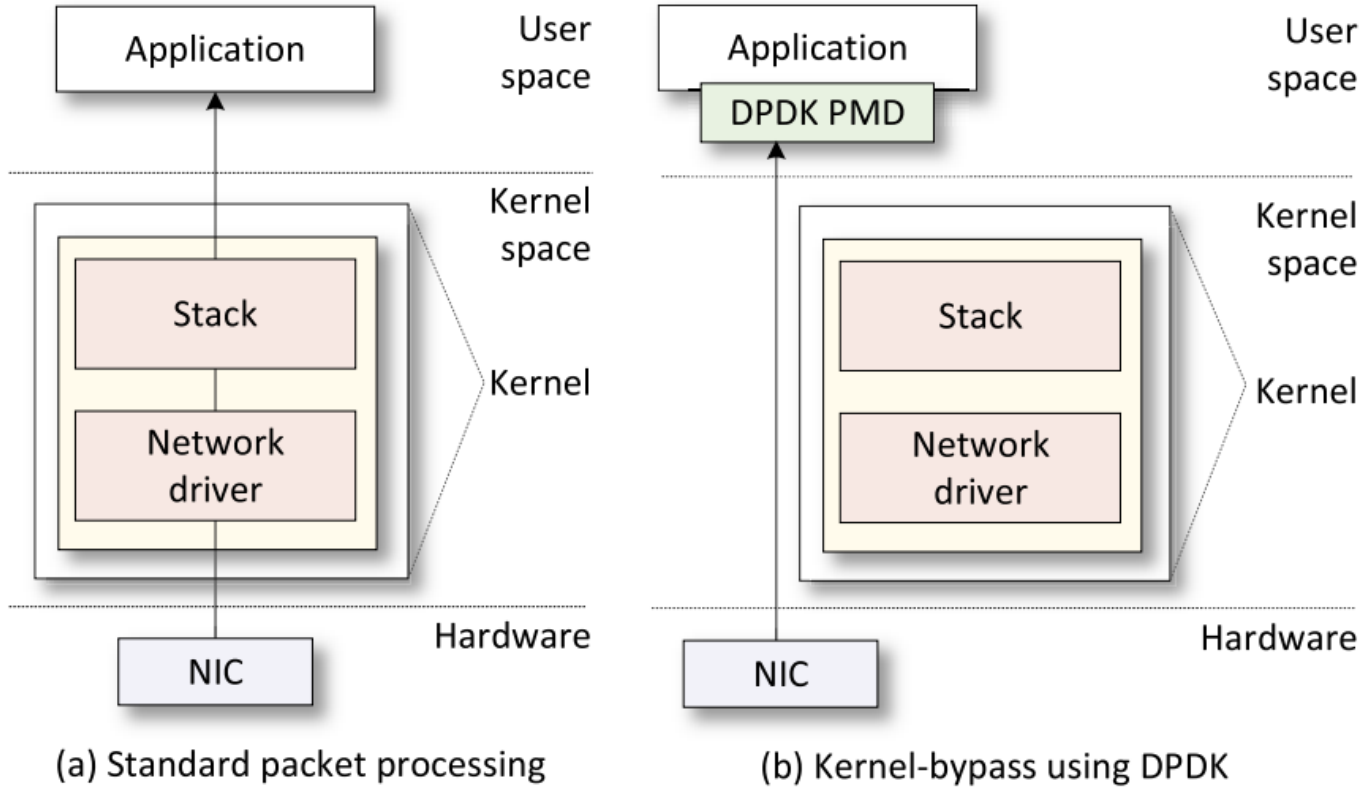
Background: PDP Switches and P4

- The programmable parser of the P4 PDP switches is the key enabler for deep packet inspection (DPI)
- The parser operates as a finite-state machine
- At each state, the parser either extracts a portion of the packet headers and stores them in pre-defined header structures or skips a pre-defined number of bits



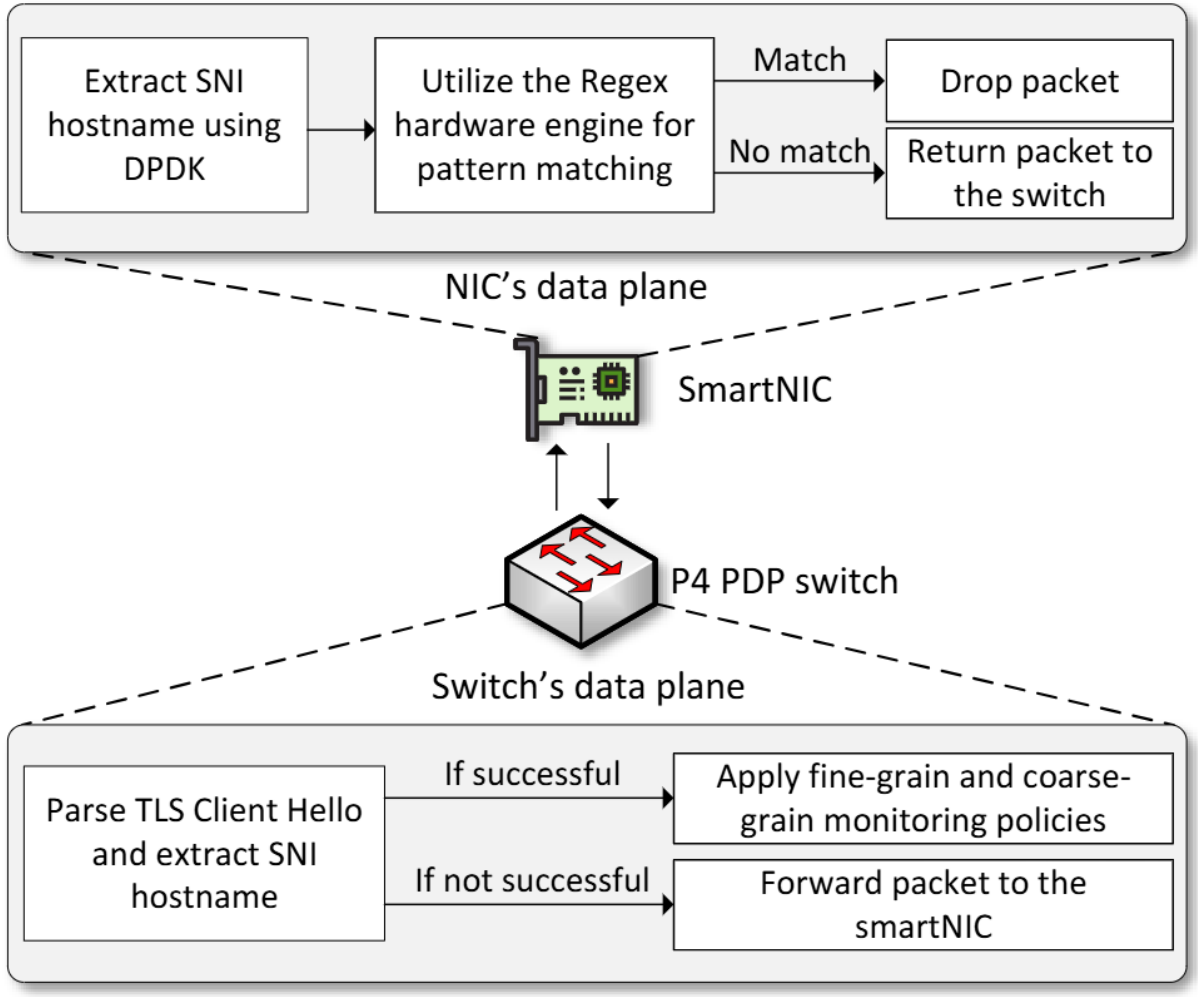
Background: SmartNICs and DPDK

- Data Plane Development Kit (DPDK) is a software-based acceleration technique
- The parser operates as a finite-state machine
- DPDK applications run on the user space and have direct access to the Network Interface Card (NIC) ports
- DPDK applications can run directly on the computing units of smartNICs



System Overview

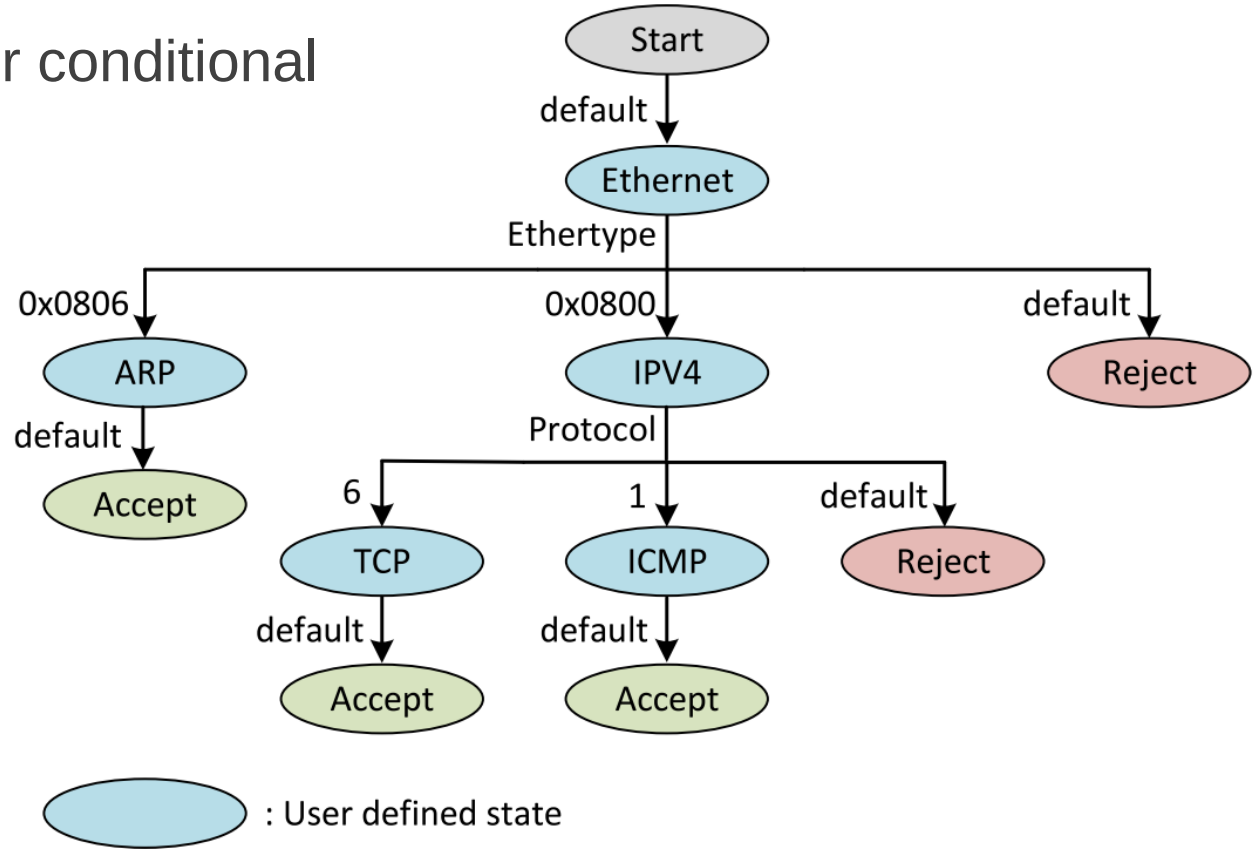
- The P4 PDP switch identifies the TLS Client Hello packets and tries to extract the SNI hostname
- If successful, monitoring and security policies are enforced on the hostname
- Otherwise, packets are forwarded to the DPDK application
- The DPDK application is capable of parsing any TLS Client Hello packet and utilizes the Regex engine to enforce coarse-grain policies



Parsing Headers in P4

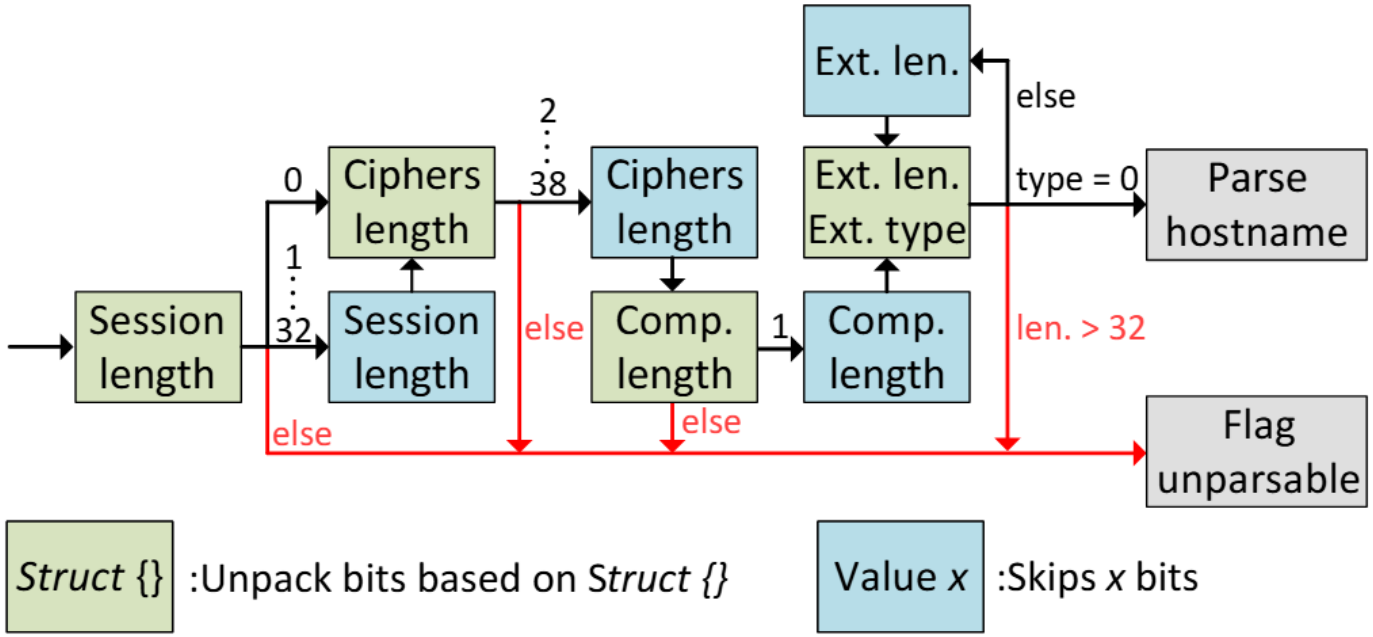
- The programmable parser is the component responsible for extracting the headers from the incoming packets
- It operates as a finite-state machine with an explicit start state, two ending states (Accept and Reject), and the user-defined states in between
- Transitioning between states can be either conditional

or unconditional



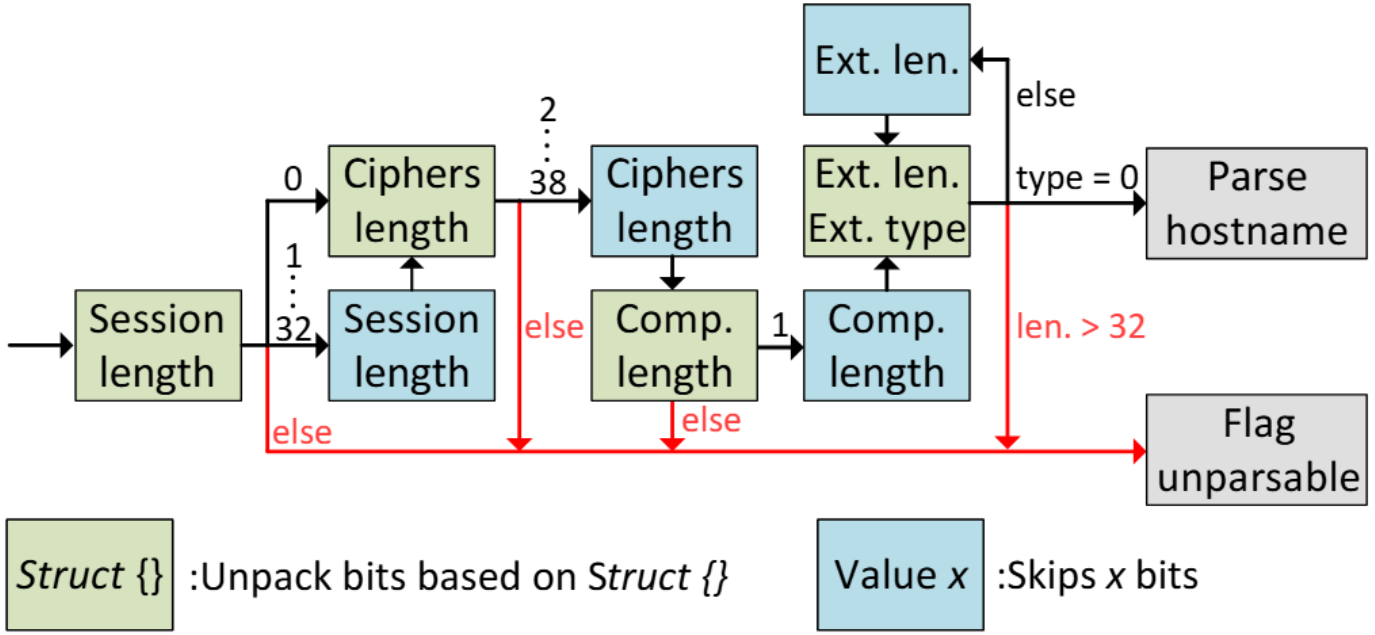
Parsing TLS in P4

- The TLS protocol contains multiple variable-length headers
- For each of these headers, the parser first extracts the length field and then uses the extracted length to transition to the corresponding state
- For each possible transition, the parser defines a separate state to parse a number of bytes equal to the length of the header



Parsing TLS in P4

- Due to hardware resource constraints, it is not feasible to account for all possible header lengths and transitions
- Instead, the parser defines states for the most commonly observed header length
- After skipping over the session ID, ciphers, and compression methods, the parser arrives at the TLS extensions
- A loop is implemented in the parser to efficiently skip over these TLS extensions



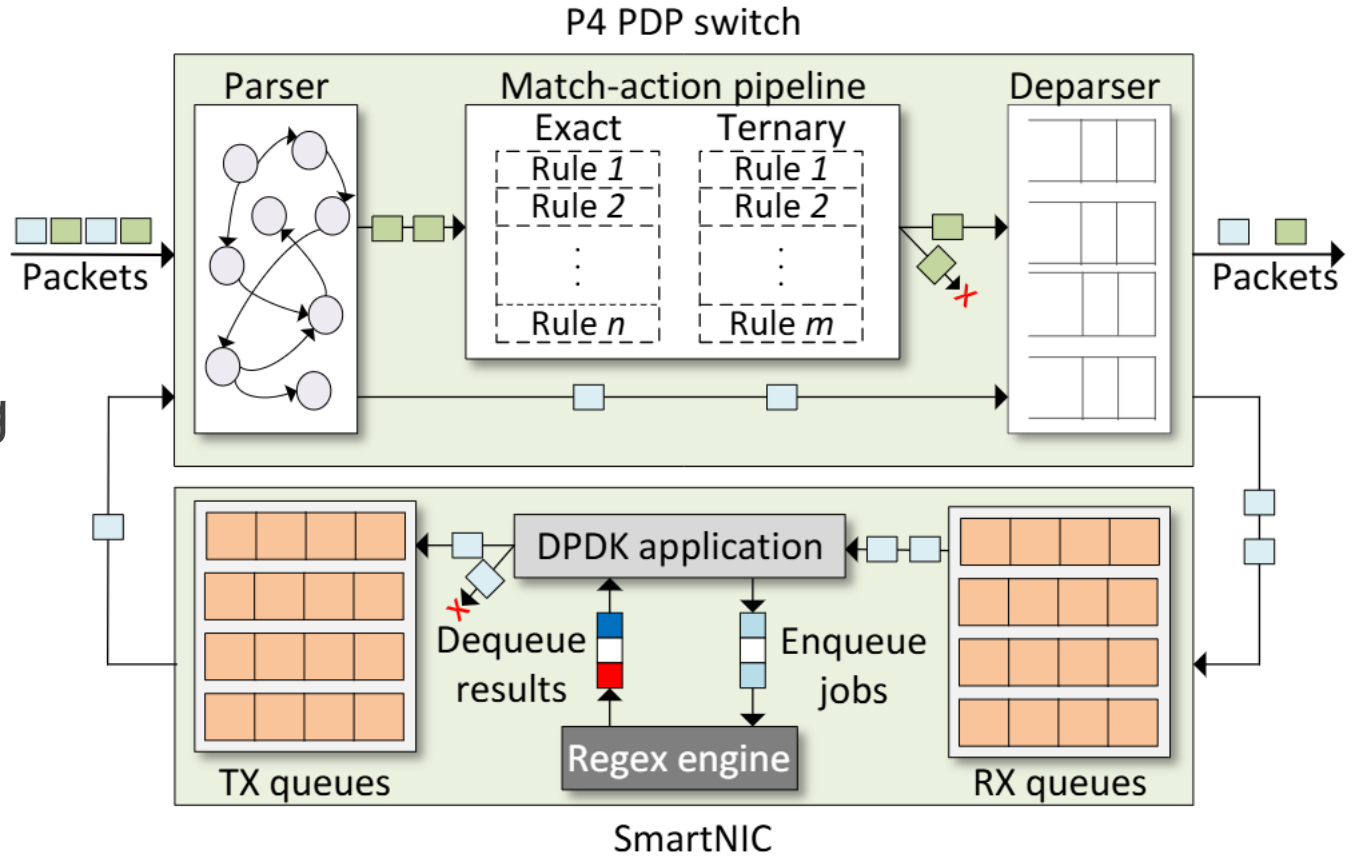
Handling Unparsed Packets in DPDK

- All packets flagged as unparsable by the P4 parser are forwarded to the DPDK application
- Unlike the P4 application, the DPDK application can handle any TLS packet without limitations on size or structure
- Similar to the P4-based processing,

the DPDK application enforces fine-grain and coarse-grain policies

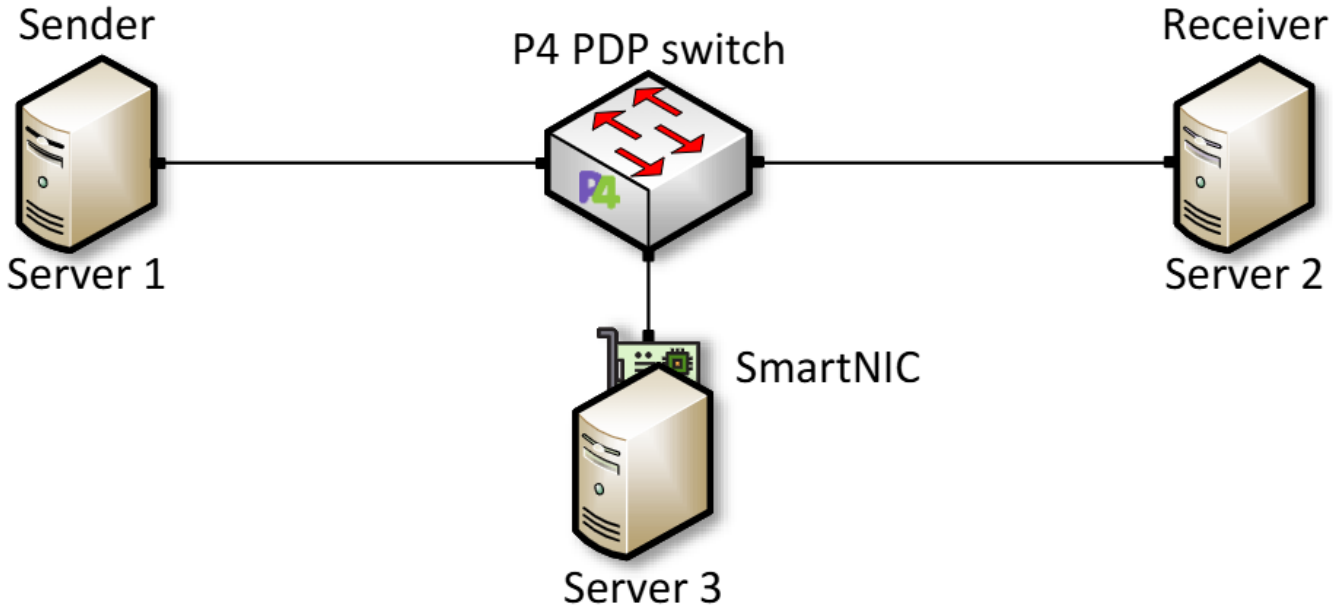
- Fine-grain policies use exact matching
- Coarse-grain policies are applied using

the smartNIC's Regex engine



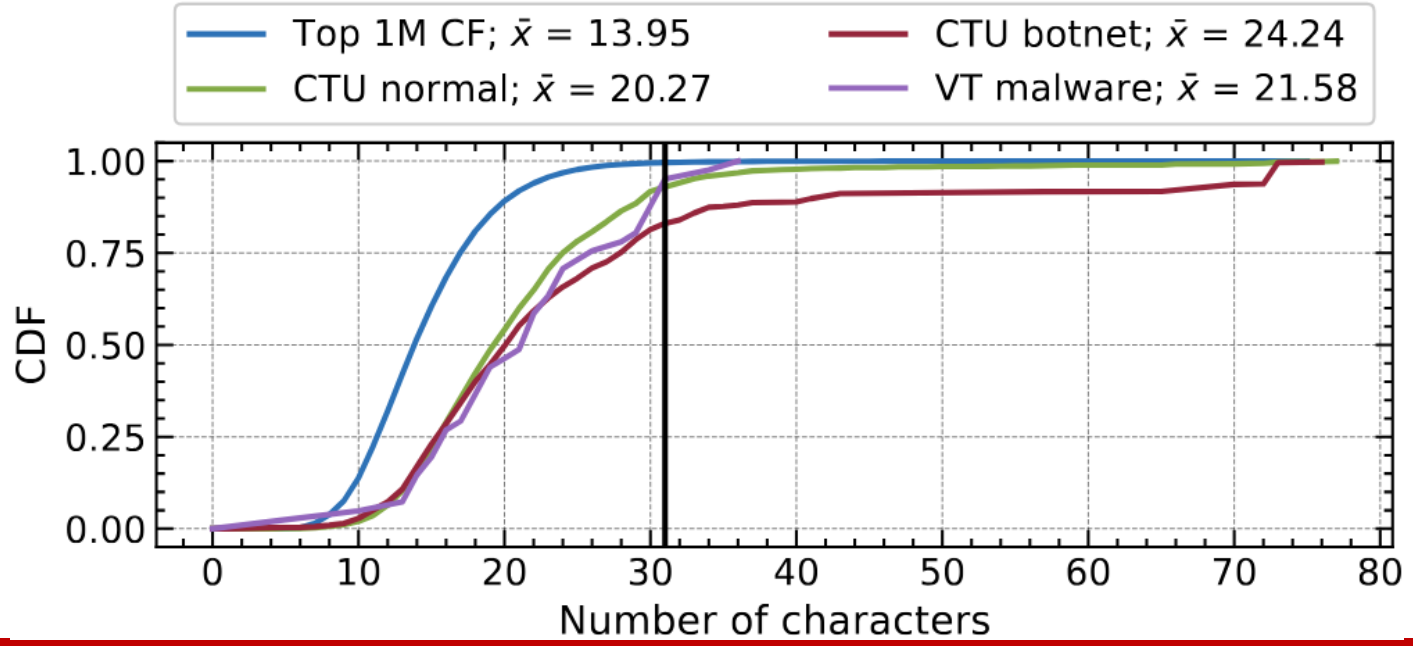
Evaluation: Experimental Topology and Datasets

- The topology used to run the evaluation experiments consists of three servers connected through a P4 PDP switch and one smartNIC
- Two servers are used to replay TLS traces from public datasets and browser traffic
- The third server is used to host the smartNIC, where the DPDK application is running
- The traces used for evaluation are mainly from Virus Total, CTU normal and botnet traffic, and Flowprint



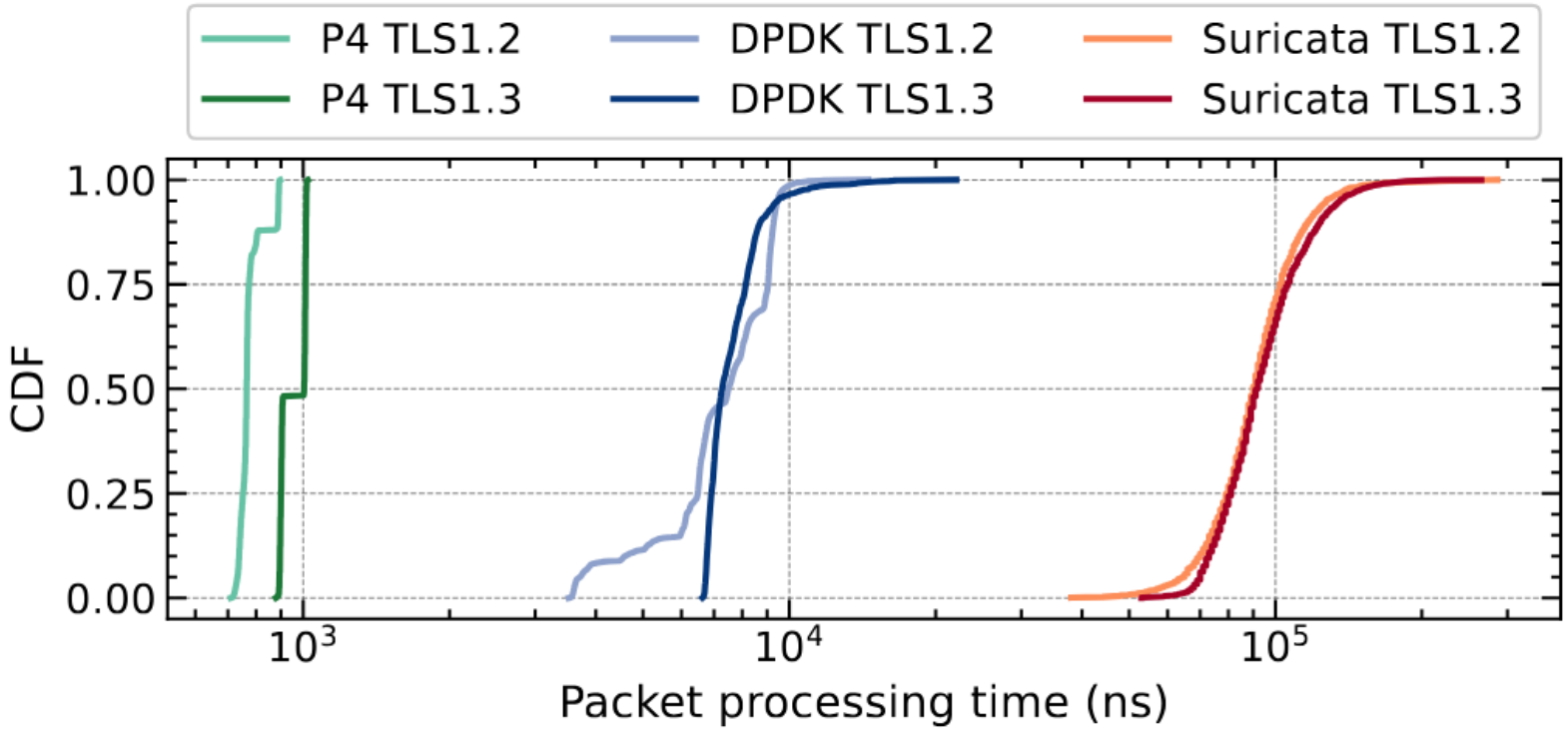
Evaluation: P4 Parsing Capabilities

- This experiment aims to identify the portion of hostnames that the P4 application can process from the collected datasets
- Among the top 1 million hostnames from Cloudflare and normal CTU traffic, 0.31% and 5.9% of hostnames, respectively, exceed the 31-character limit and are therefore not parsable in P4
- For malicious hostnames from the CTU botnet and Virus Total,15.9% and 4.8% of hostnames, respectively, are excluded for the same reason



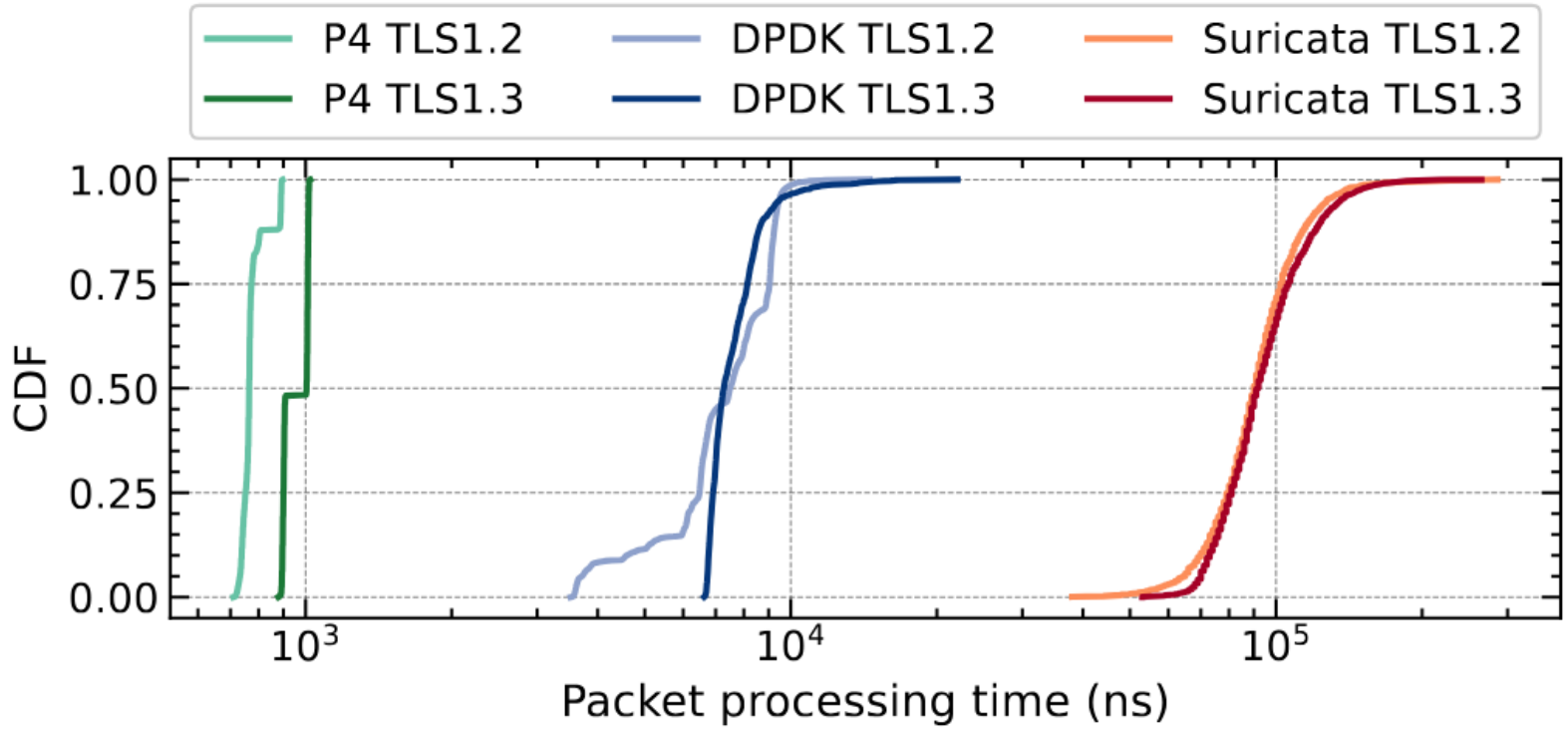
Evaluation: Comparing the proposed system with Suricata

- To evaluate the performance gain of the proposed system, it is compared to the Suricata Intrusion Detection/Prevention System(IDS/IPS)
- Suricata is running over a 32-core general-purpose CPU
- The P4 switch requires, on average, 700 nanoseconds (ns) and 900 ns to process TLS1.2 and TLS1.3 packets, respectively



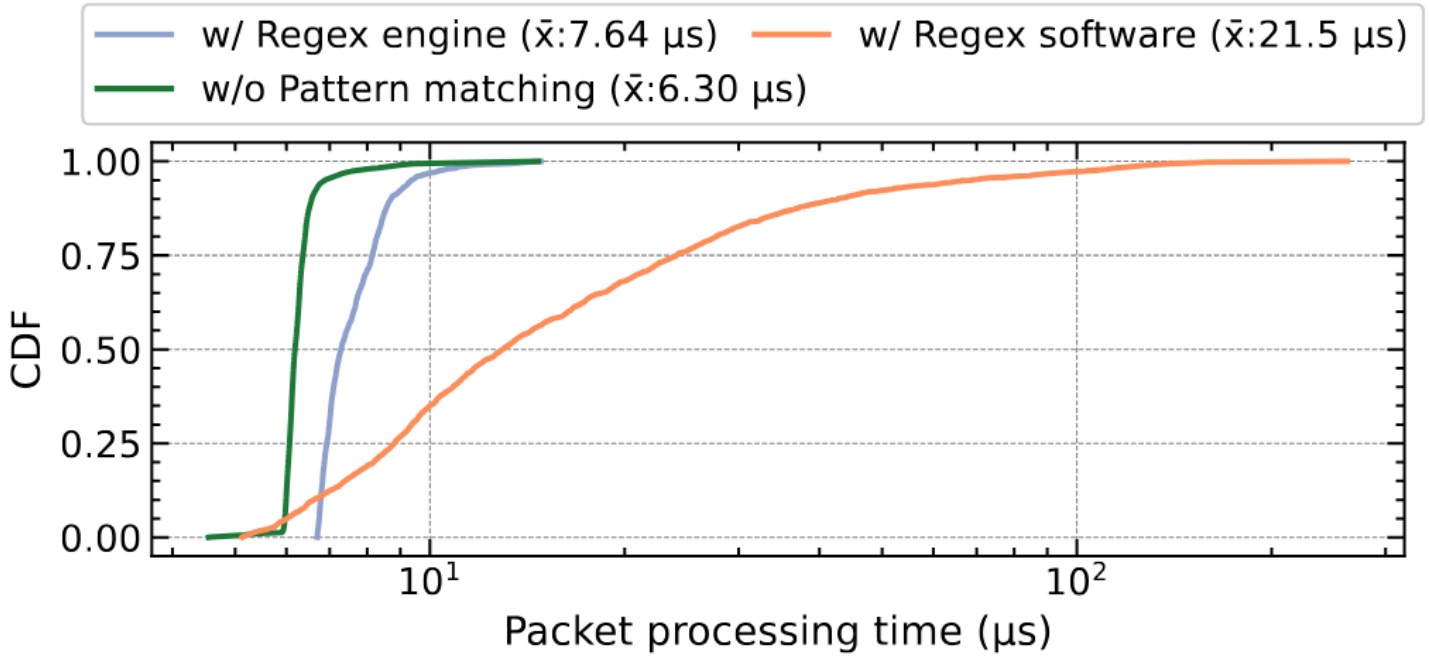
Evaluation: Comparing the proposed system with Suricata

- The DPDK application requires, on average, 7.2 microseconds (µs) and 7.5 µs to process TLS1.2 and TLS1.3 packets, respectively
- Suricata requires, on average, 92 µs and 95 µs to process TLS1.2 and TLS1.3 packets, respectively



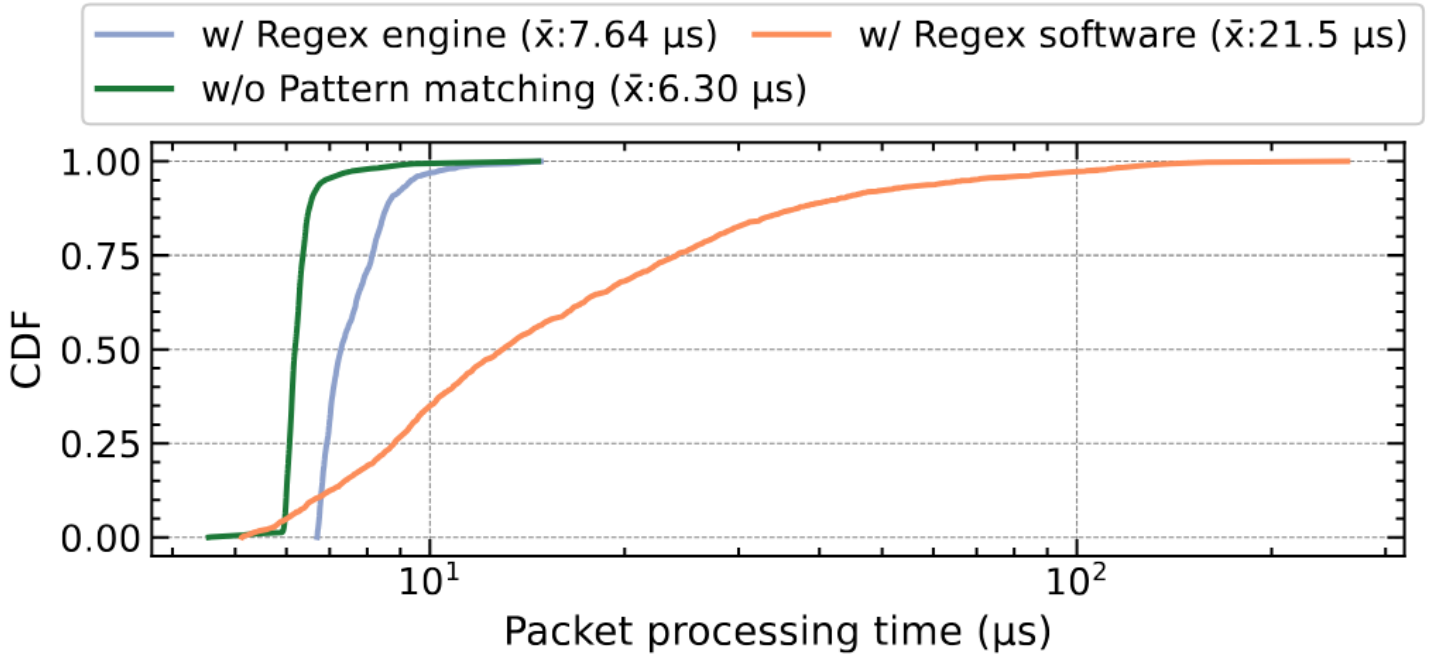
Evaluating the Regex engine in the proposed system

- All the evaluations on the DPDK application show its performance by running on a single core of the smartNIC
- For stress testing, TLS packets are forwarded to the DPDK application at 1.5 Gbps
- Without pattern matching, DPDK requires, on average, 6.3 µs to process TLS packets



Evaluating the Regex engine in the proposed system

- Enabling the pattern matching feature using the Regex engine of the smartNIC increases the average processing time by 1.34 µs to 7.64 µs
- Implementing pattern matching using the Regex software (i.e., using the Regex library) increases packet processing time by 15.2 µs be 21.5 µs



Acknowledgement

- This work was supported by the U.S. National Science Foundation (NSF), under awards 2403360, 2346726