# Unlocking Network Visibility through Programmable Data Planes

Gerrit Avenant
April 2025

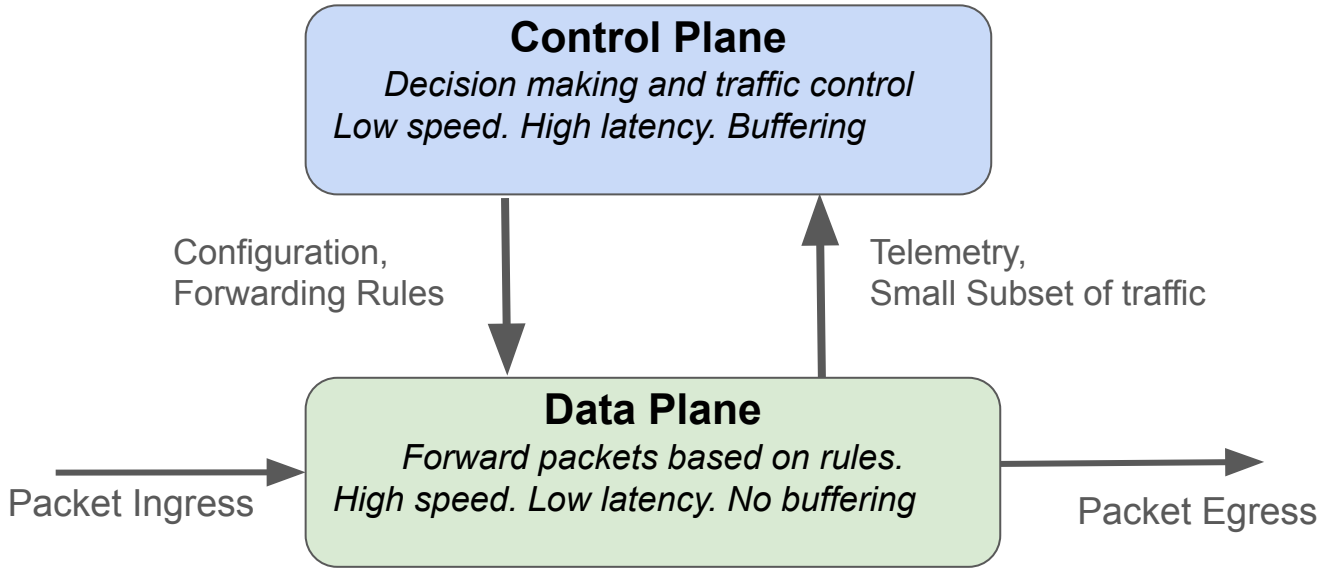
# Introduction

- Electronic/Computer Engineer at **Dendrite Cyber**
- Specialize in **customised packet processing solutions**
- Focus areas:
  - Improved visibility for high speed networks
  - Network traffic health metrics

# What We'll Cover

- **Introduction to Programmable Data Planes**
  - Overview of Hardware
  - Software Libraries & Languages
- **Network Monitoring Use Cases**
  - MPLS Decapsulation
  - Packet Deduplication

# Terminology



# Fixed Function Network Devices

- **Standard** network components
  - NICs, Switches, Routers, Firewalls, Packet Brokers
- **Hardware Based** Processing (ASICs)
- **Cost Effective**

- Limitations:
  - **Vendor-defined** features
  - **Not flexible** in terms of packet processing
  - **Limited** SDN **integration**
  - **No** support for **new/custom protocols**

# Programmable Dataplane Technologies

Driven by Modern Data Center Architectures:

- **East-West traffic increased**
- **Smart data movement** required

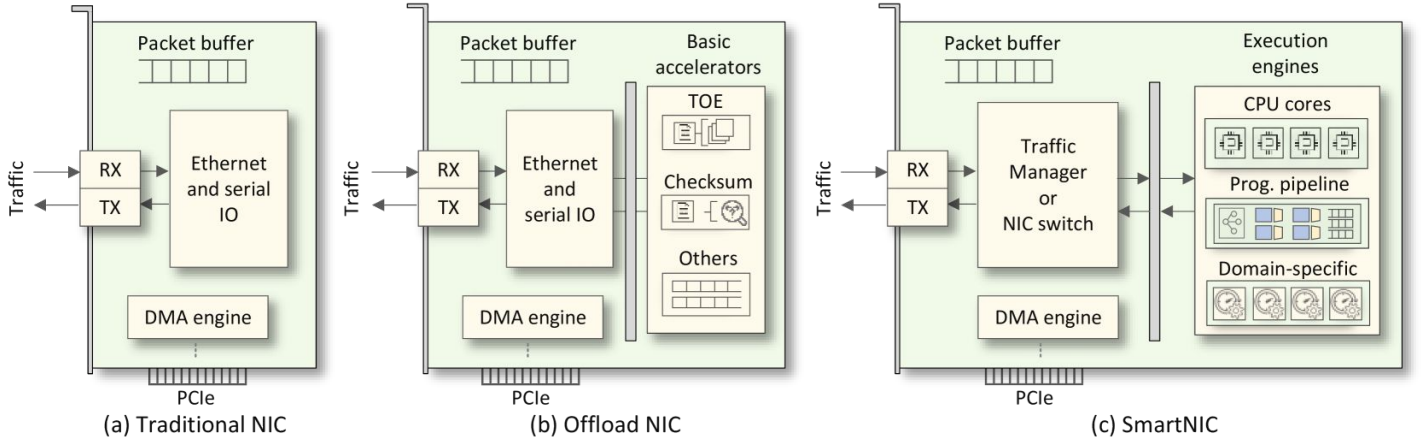
Programmable hardware allows:

- Custom hardware **offloading**
- **Multi-purpose use** of device

# Hardware: Network Accelerator Cards

- **PCIe-based** cards

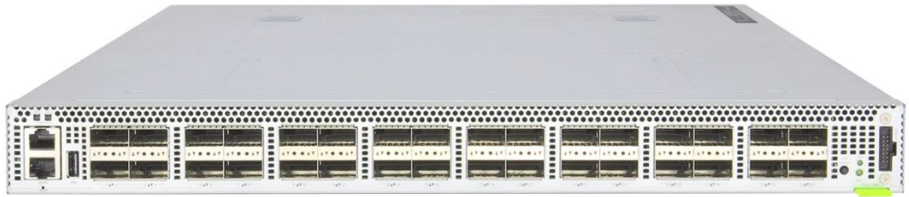
*arXiv:2405.09499v1*




- **Offload** CPU Intensive tasks to SmartNIC
  - Header Parsing,
  - Encryption/Decryption
  - Storage Offload

# Hardware: Programmable Switches

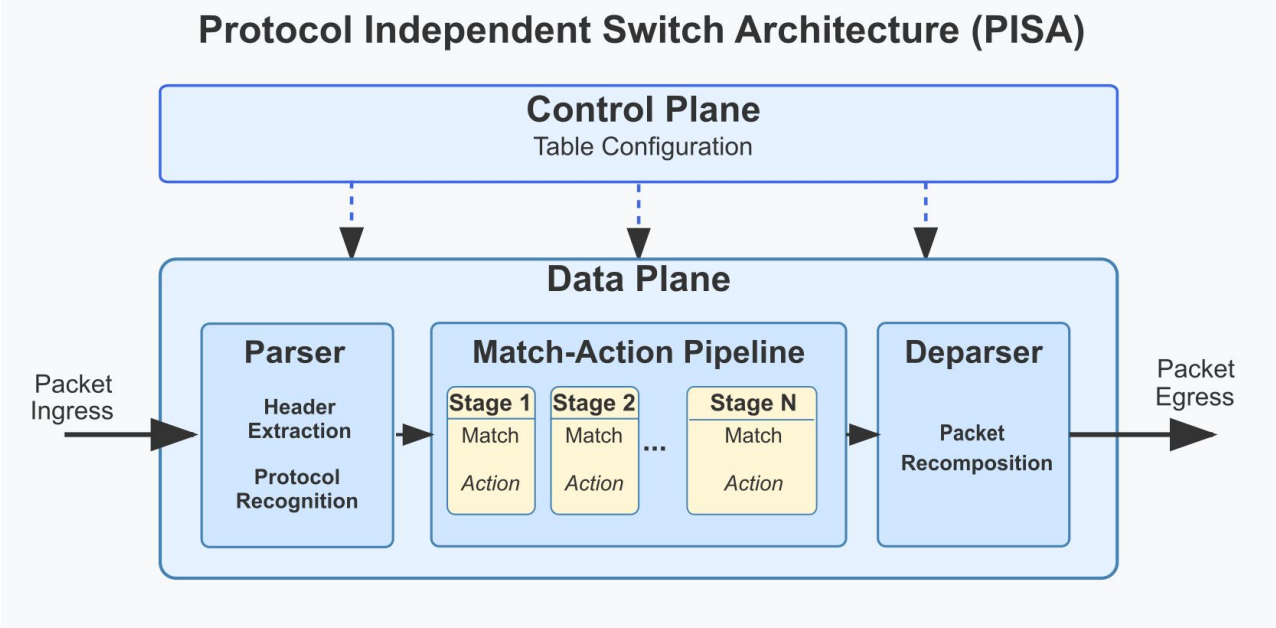
- Intel Tofino (EOL)
- **Fully programmable** packet processing pipeline (**P4** language)



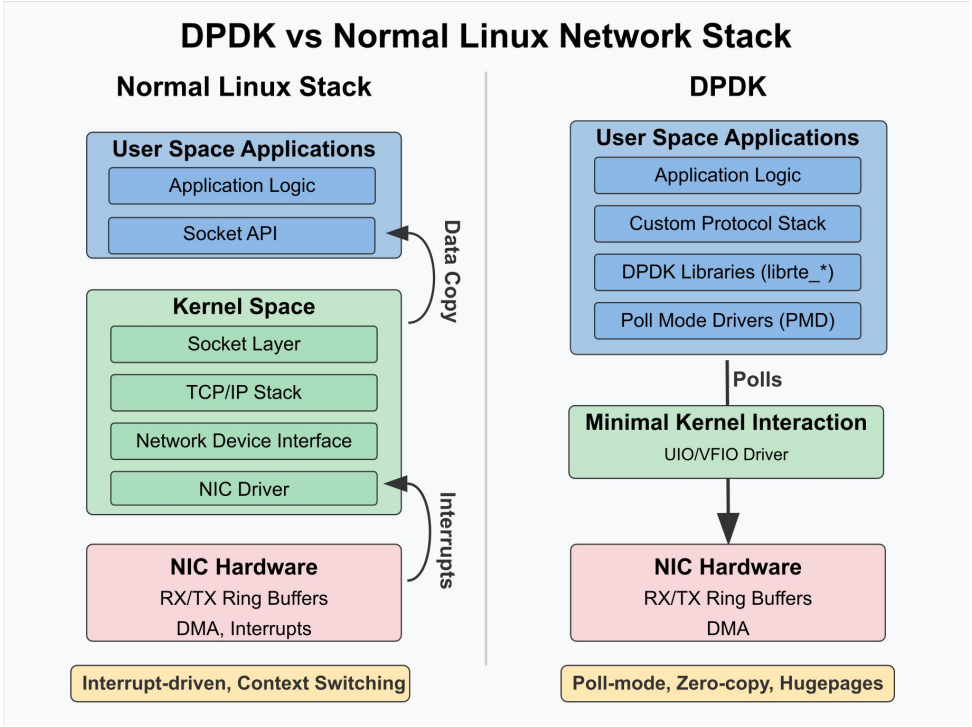
- **3.2 Tb/s** Tofino Programmable Pipeline (Data Plane)
- **4-core** Intel® Pentium® D-1517  (Control Plane)
- **32x 100Gbps** QSFP Ports

# Software: P4 Language

- “**P**rogramming **P**rotocol-independent **P**acket **P**rocessors”
- **Targets** - Programmable Switches, SmartNICs, XDP, eBPF



# Software: DPDK (Dataplane Development Kit)

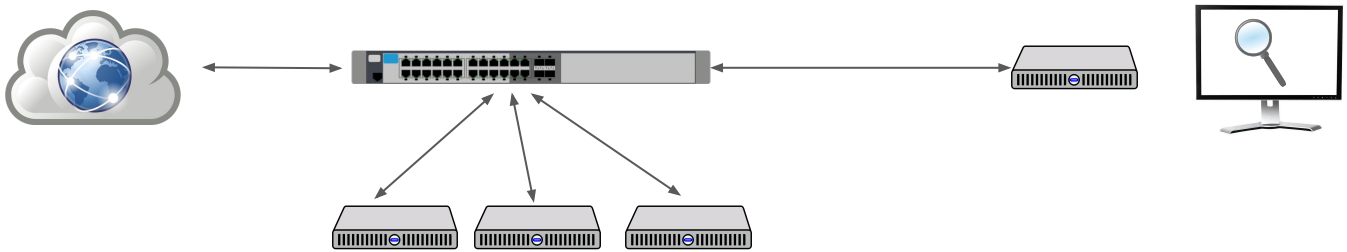



- User Space C++ Library
- **Bypass kernel** network stack
- Optimised for **standard NICs**

- **Linux Kernel**
  - ~1Gbps per CPU core
- **DPDK**:
  - ~10Gbps per CPU core

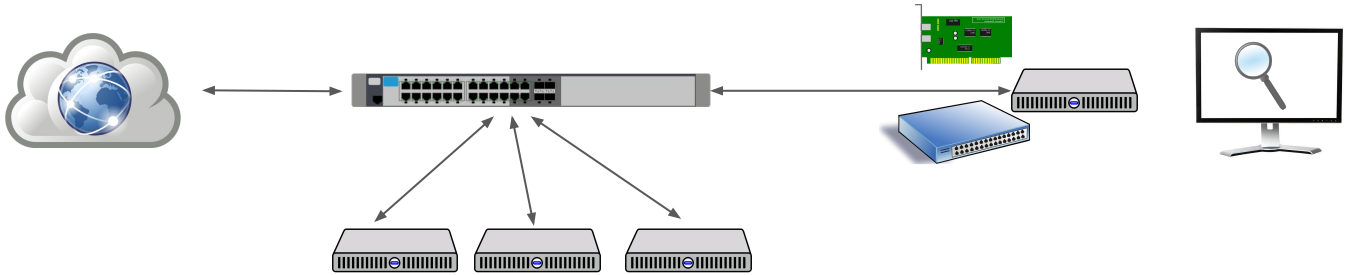
# Network Monitoring Applications

- Growing demand for **real-time analytics**
- Most **monitoring tools** are **software-based** and will have performance limits



# Network Monitoring Applications

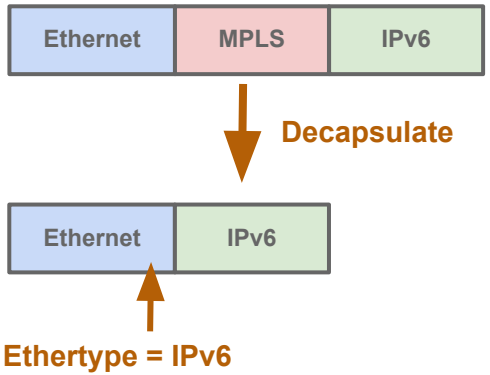
- Growing demand for **real-time analytics**
- Most **monitoring tools** are **software-based** and will have performance limits



- ***“Do what you can in hardware, do what you must in software.”***

# Case Study: MPLS Decapsulation (1)

- **Goal**: Remove MPLS headers for downstream compatibility
- **Challenge**: MPLS header doesn't indicate next proto/ethertype!!!



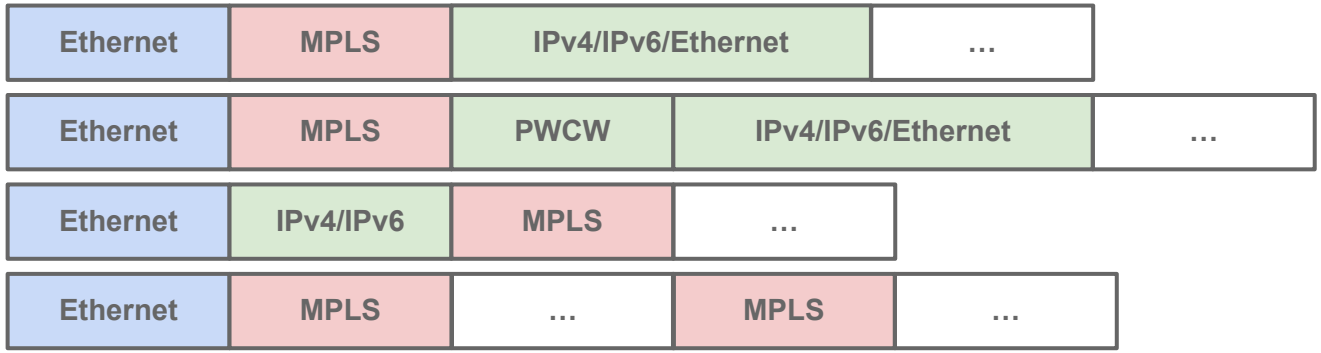

Most solutions use "first nibble hack":
First nibble after MPLS:

| | |
|---|---|
| **4**: | IPv4 header |
| **6**: | IPv6 header |
| **0**: | PCWC |
| **Rest**: | Ethernet |

It works *most* of the time….

# Case Study: MPLS Decapsulation (2)

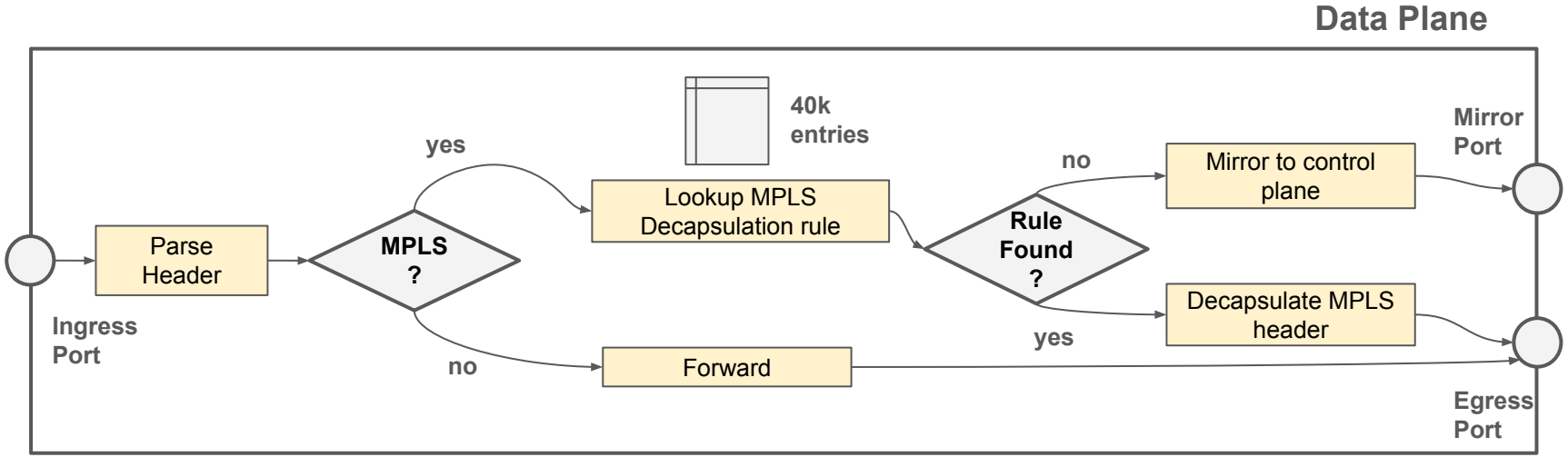
- Header combinations:




- Other Challenges:
  - **Nested** MPLS headers
  - **Multiple** Labels (7+)
  - **Fragmented** MPLS-over-IPv4

# Case Study: MPLS Decapsulation (3)

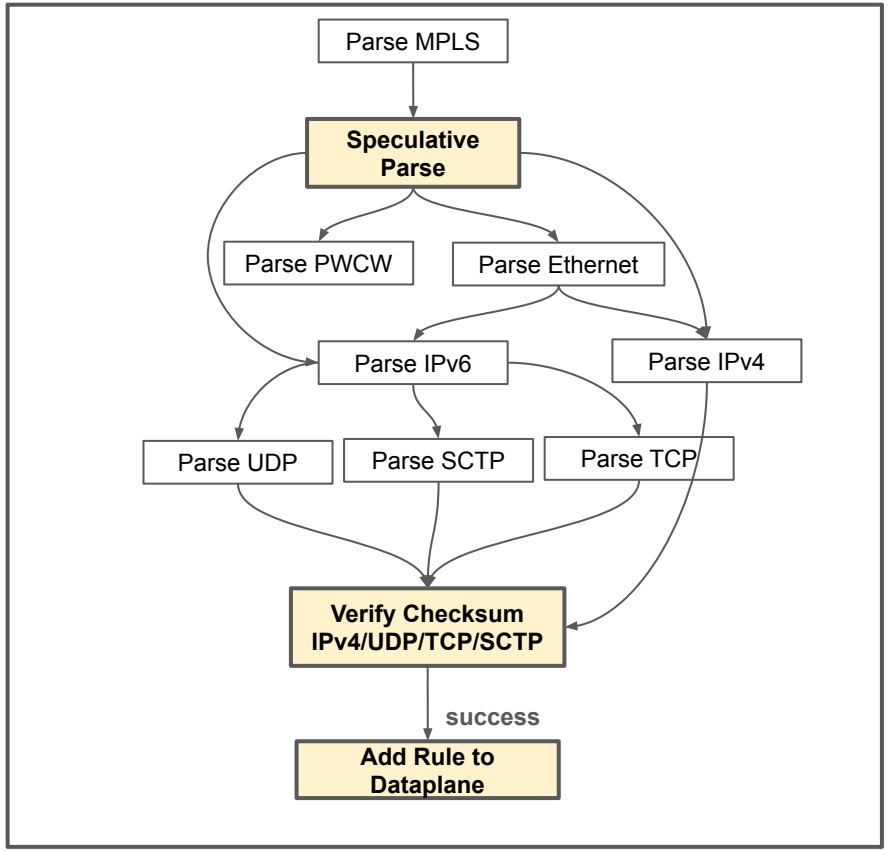
- **Solution**:
  - Use Programmable Hardware
  - Decapsulate flows in data plane
  - Learn decapsulation rules through speculative parsing in control plane



# Case Study: MPLS Decapsulation (4)

Speculative parsing in Control Plane

- See what header structure fits
- Add dataplane rule if checksum validation succeeds



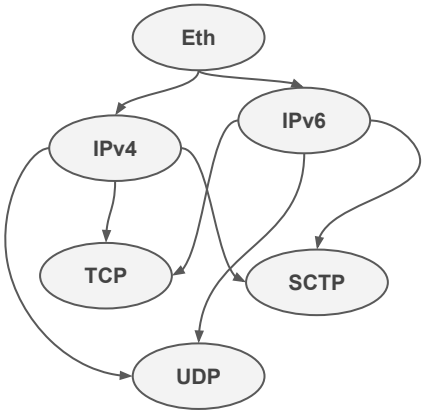
# Case Study: Deduplication (1)

- **Challenge**: Traffic from several points in the network is sent to monitoring tools
- **Goal:** Reduce processing load on downstream tools through deduplication

- Many products perform deduplication in **software** on a CPU
- Software-based solutions are **resource-intensive** and introduce **latency**

# Case Study: Deduplication (2)

Parse Packet Headers




Calculate Hash over Packet Fields

**IPv4 Header**

| |
|---|
| Version \| IHL \| Type of Service |
| **Total Length** |
| **Identification** |
| Flags \| Fragment Offset |
| Time to Live \| Protocol |
| **Header Checksum** |
| Source IP Address |
| Destination IP Address |

**TCP Header**

| |
|---|
| **Source Port** |
| **Destination Port** |
| **Sequence Number** |
| Acknowledgment Number |
| Data Offset \| Reserved \| Flags |
| Window Size |
| **Checksum** |
| Options \| Padding |

**Hash Calculation**

Store in Table

| | |
|---|---|
| 0 | 622a37e4964eb3e1693add76e77653ef |
| 1 | 268429625d86f29ac15ccd6b8b8c77c5 |
| 2 | |
| 3 | 952a52f9b0acbbc82e832c3dd749fa54 |
| 4 | e19b881f9c9228ac78f3fff69ce53e99 |
| 5 | |
| 6 | a87cf53f40696787017fdb0fa8a074bf |

**Index Collision**

Packet ID matched?
Packet within in time window?

**YES**, **DROP**!!

# Case Study: Deduplication (3)

- **Solution**:
  - Use Programmable **Hardware**:
  - Use **Programmable Parser** to parse headers at **line rate**
  - **Hash Calculation** over several fields to uniquely identify packet
  - Use Registers for **Hash Table lookups** and Timestamp Comparison
  - Registers (**Memory**) physical limitation to how many packets can be stored at once
- **Results**:

| Traffic | Packet Size | Window | Traffic Dropped |
|---|---|---|---|
| 1 x 100Gbps | 300 | 1ms | 100% |
| 5 x 100Gbps | 300 | 1ms | 99.5% |

# Overview

**Benefits** of Programmable Devices for Monitoring Use Cases:

- **Reduces** CPU **overhead** by offloading intensive packet processing tasks.
- Enables **flexible** network packet processing and **customised** solutions to challenges
- Improves **efficiency of network monitoring** at high network rates.