eBPF AND XDP FOR HIGH-PERFORMANCE NETWORK PROGRAMMING

Syeda Zunehra Banu, Christian Tran



MILESTONE #2 TOPICS CONSIDERED

eBPF / XDP

Early programmable packet processing

SmartNICs

Programmable network offloading

Choreographic Programming

Endpoint Projection

Distributed Object Communication

Communication among distributed objects

CHOSEN TOPIC: eBPF / XDP

WHY WE CHOSE eBPF / XDP



SOFTWARE-BASED APPROACH

Focuses on programmable packet processing in Linux rather than specialized networking hardware.

EARLY PACKET PROCESSING

Explores how packets can be handled before much of the normal Linux networking stack.

REAL WORLD USES

Provides concrete examples such as DDoS mitigation, filtering, and monitoring.

We selected eBPF/XDP because it offers a software-based approach to high-performance networking with practical applications.

Key benefits of eBPF / XDP



- **PERFORMANCE**

Networks must process large volumes of traffic quickly

PROGRAMMABIILITY

eBPF/XDP lets developers customize packet handling in Linux.

REAL WORLD USES

Applications include filtering, monitoring, security, and traffic management

WHY THIS MATTERS

- Packets may require inspection or a forwarding decision.
- Traditional processing can involve multiple layers of the OS networking stack.
- Extra processing can add overhead, especially when traffic volume is high
- Security and monitoring systems need timely packet decisions.

- The challenge
  - Process useful traffic efficiently while identifying unwanted or suspicious packets.
- The opportunity
  - Make selected packet decisions earlier, before packets travel further through the stack.

THE INDUSTRY TREND: PROGRAMMABLE DATA PLANES

Traditional networking

Mostly fixed processing behavior

→

Programmable networking

Customizable packet processing

Industry trend: greater flexibility and programmability

- Network operators want to customize packet processing without relying on vendor firmware changes.
- eBPF and XDP support programmable packet processing in Linux.

WHAT IS eBPF?

eBPF = Extended Berkeley Packet Filter

eBPF allows developers to run custom programs inside the Linux kernel without modifying the kernel itself.

PROGRAMMABLE

Runs verified bytecode at supported Linux kernel hook points.

FLEXIBLE

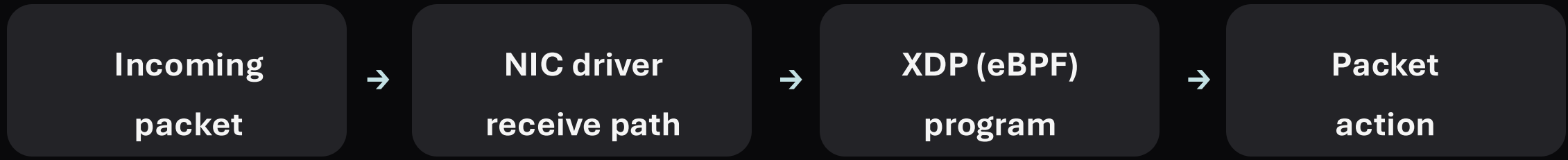
Can modify packet processing and exchange data with user space.

eBPF programs can be loaded and replaced without rebuilding the Linux kernel.

WHAT IS XDP?

XDP = eXpress Data Path

XDP is a Linux networking technology that uses eBPF programs to process incoming packets early, before most of the normal networking stack.



TRADITIONAL PROCESSING VS. EARLY PACKET DECISIONS

TRADITIONAL PATH

Incoming packet

↓

Multiple networking layers

↓

filtering or handling

More of the stack may be involved before a decision is made

WITH XDP

Incoming packet

↓

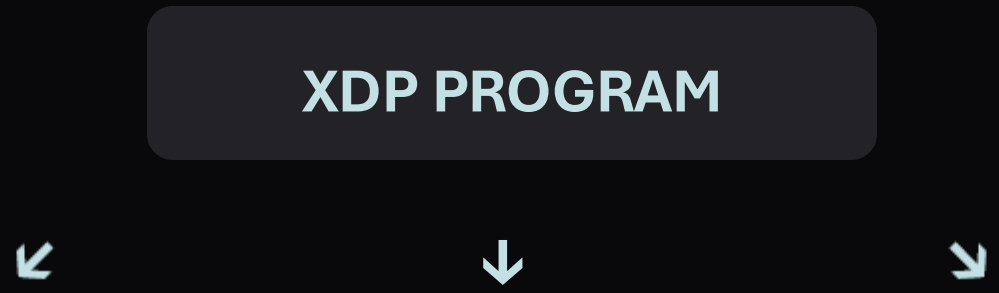
XDP decision

↓

Drop, pass or redirect

Unwanted packets can be discarded before additional processing

HOW XDP DECISIONS WORK



DROP

Discard unwanted packets early.

PASS

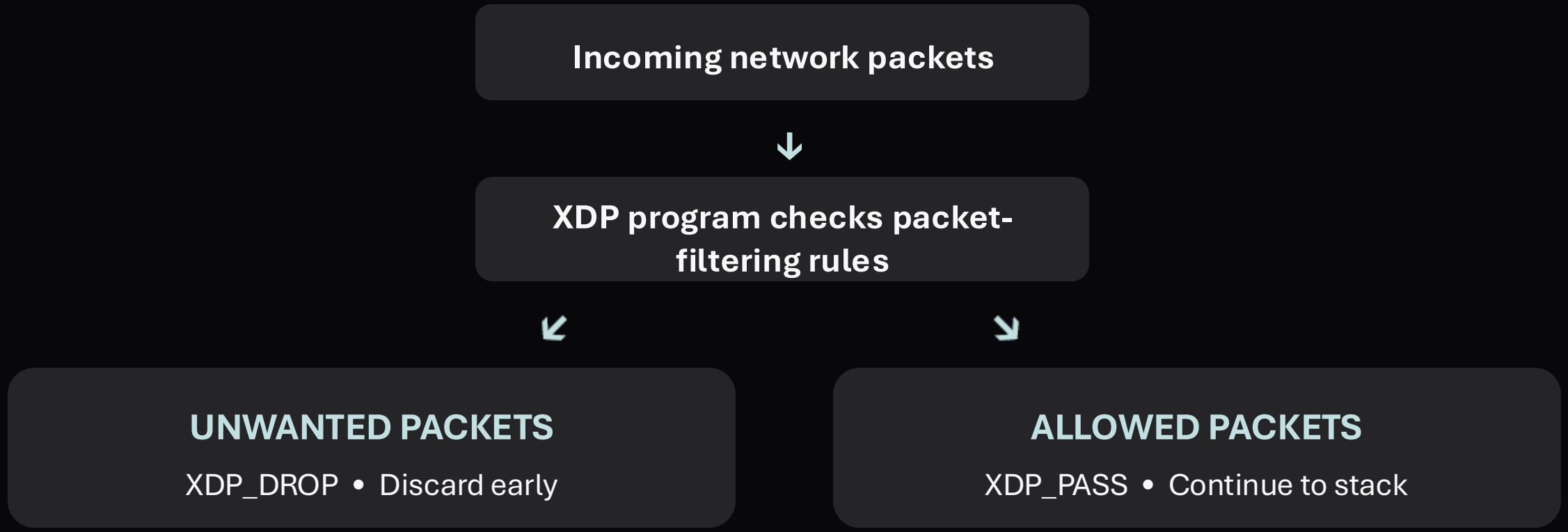
Continue through the normal kernel stack.

REDIRECT

Send to another supported interface or destination.

REAL-WORLD EXAMPLE: DDoS MITIGATION

Illustrative packet-filtering scenario



Benefit: unwanted packets can be rejected before further networking-stack processing.

WHERE CAN eBPF / XDP BE APPLIED?

DDoS MITIGATION

Drop selected malicious or unwanted packets early.

MONITORING

Gather packet count, traffic patterns, system performance

TRAFFIC MANAGEMENT

Filter, forward or redirect traffic based on rules.

Common benefit: programmable packet processing for performance, security, and visibility.

WHY THIS MATTERS TO NETWORK PROGRAMMERS

- More control over how network traffic is handled.
- No longer need to rely entirely on the traditional networking process.
- Choose whether packets continue, are discarded, or are redirected.
- Useful for network security tools, packet filters, and traffic management.

Key takeaway: programmability + early packet decisions.

Early Packet Filtering logic with eBPF / XDP

```
// After validating Ethernet + IPv4 headers:
if (ip->saddr == bpf_htonl(0xC000020A))
    return XDP_DROP;  // Block 192.0.2.10

return XDP_PASS;      // Pass other packets
```

BLOCKED SOURCE IP

192.0.2.10

XDP_DROP → Discard early

OTHER SOURCE IPS

XDP_PASS

Continue through the normal Linux networking stack

REFERENCEs and QUESTIONS

1. **Brandino, B., & Grampín, E. (2024).**

Network Data Plane Programming Languages: A Survey.

Computers, 13(12), Article 314.

https://doi.org/10.3390/computers13120314

2. Red Hat Developers. (2021).

Get Started with XDP.

https://developers.redhat.com/blog/2021/04/01/get-started-with-xdp

Thank you! Questions?