

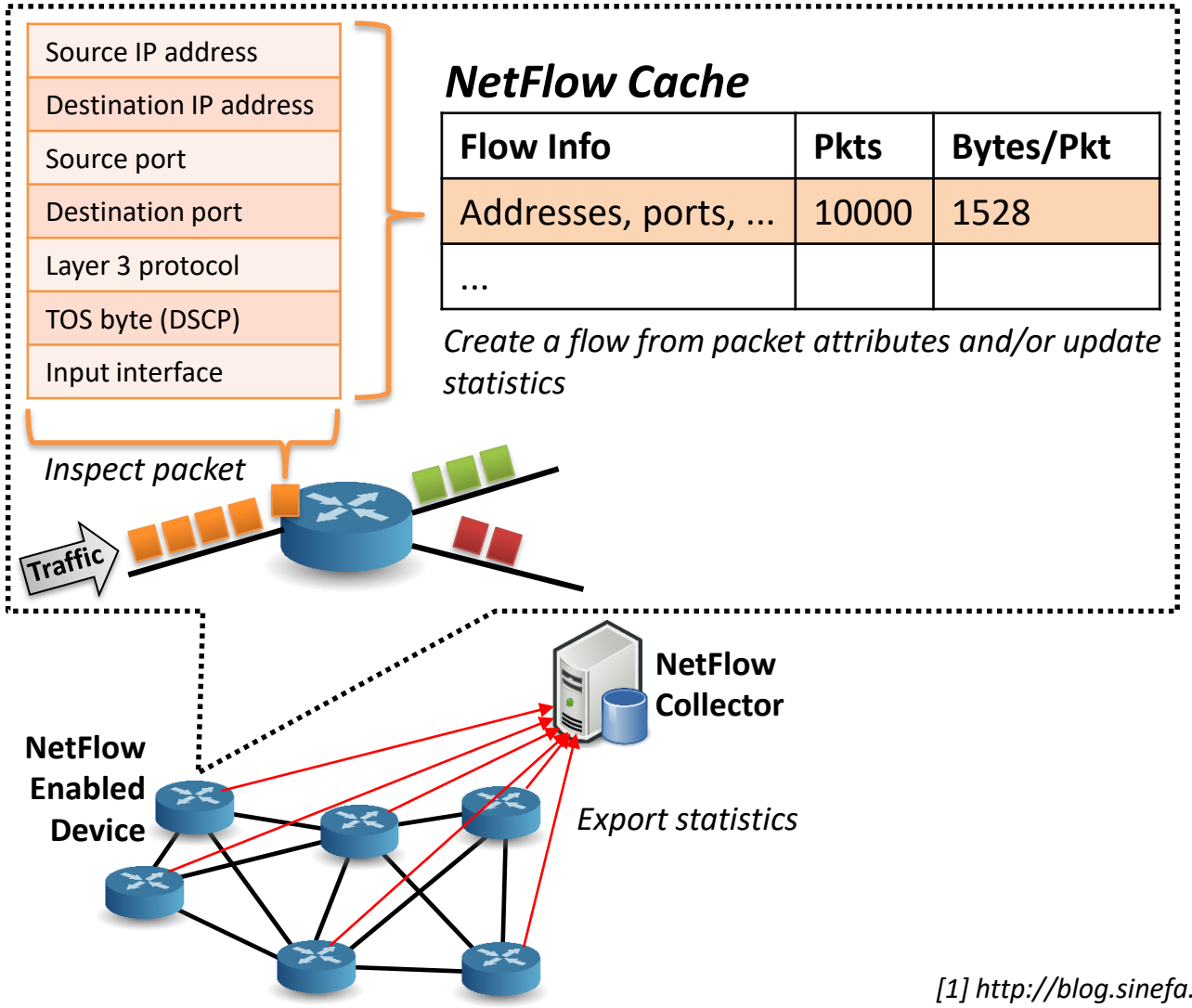

Empowering network monitoring through programmability

Marco Savi, Roberto Doriguzzi Corin, Domenico Siracusa
Fondazione Bruno Kessler (FBK) - CREATE-NET Research Center

FONDAZIONE BRUNO KESSLER | CREATE-NET



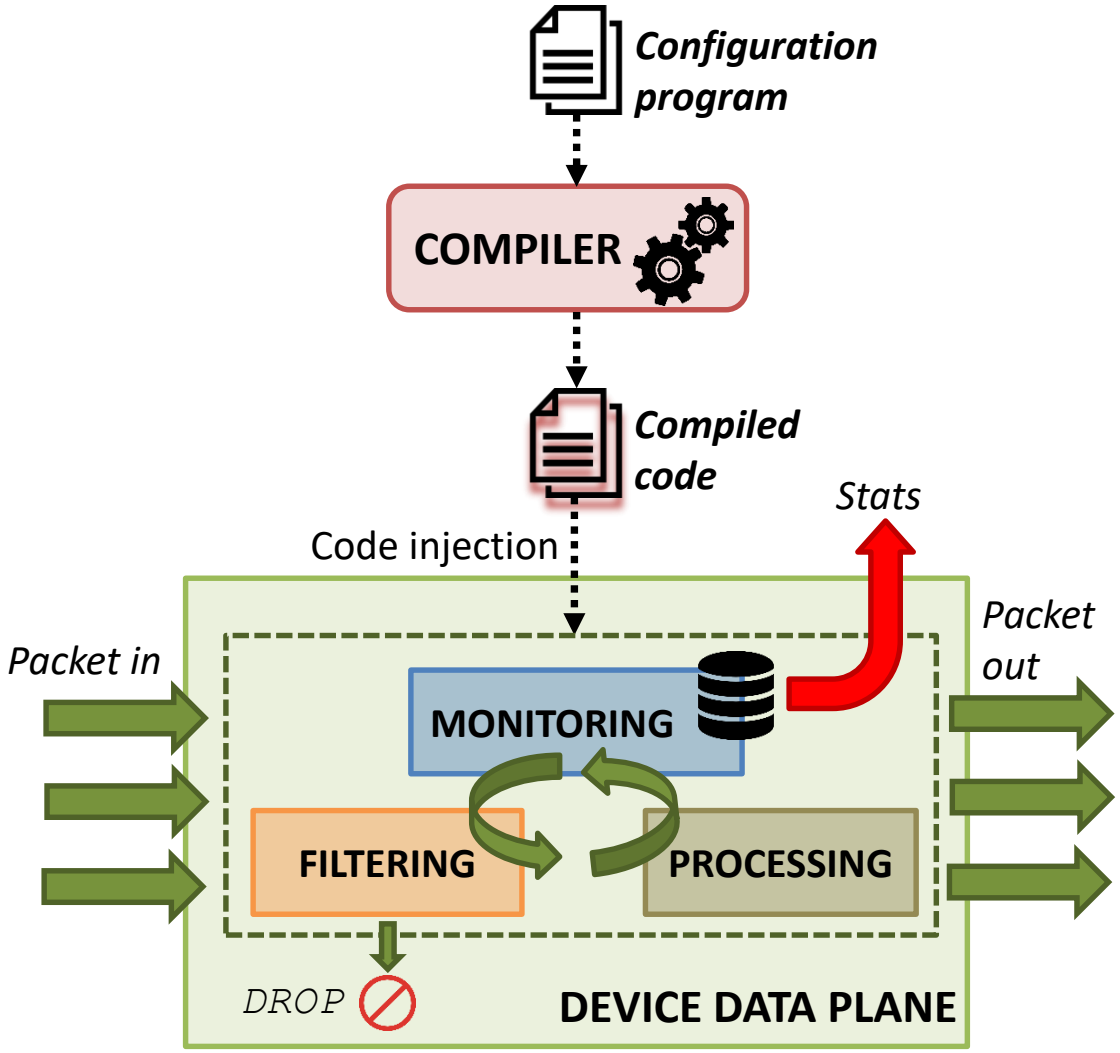

Passive Network Flow Monitoring – Current approach



- **NetFlow**: most well-known network flow monitoring protocol
  - Others exist (e.g. **sFlow**), similar drawbacks
- Well-known **drawbacks** [1]
  - High load on hardware (i.e., CPU, storage, memory) equipment
  - Low time granularity (typically 5 mins) to avoid high network overhead
- **Sampling** needed to overcome performance issues
  - Reduced accuracy!
- **Programmable data-planes** can help improve network flow monitoring

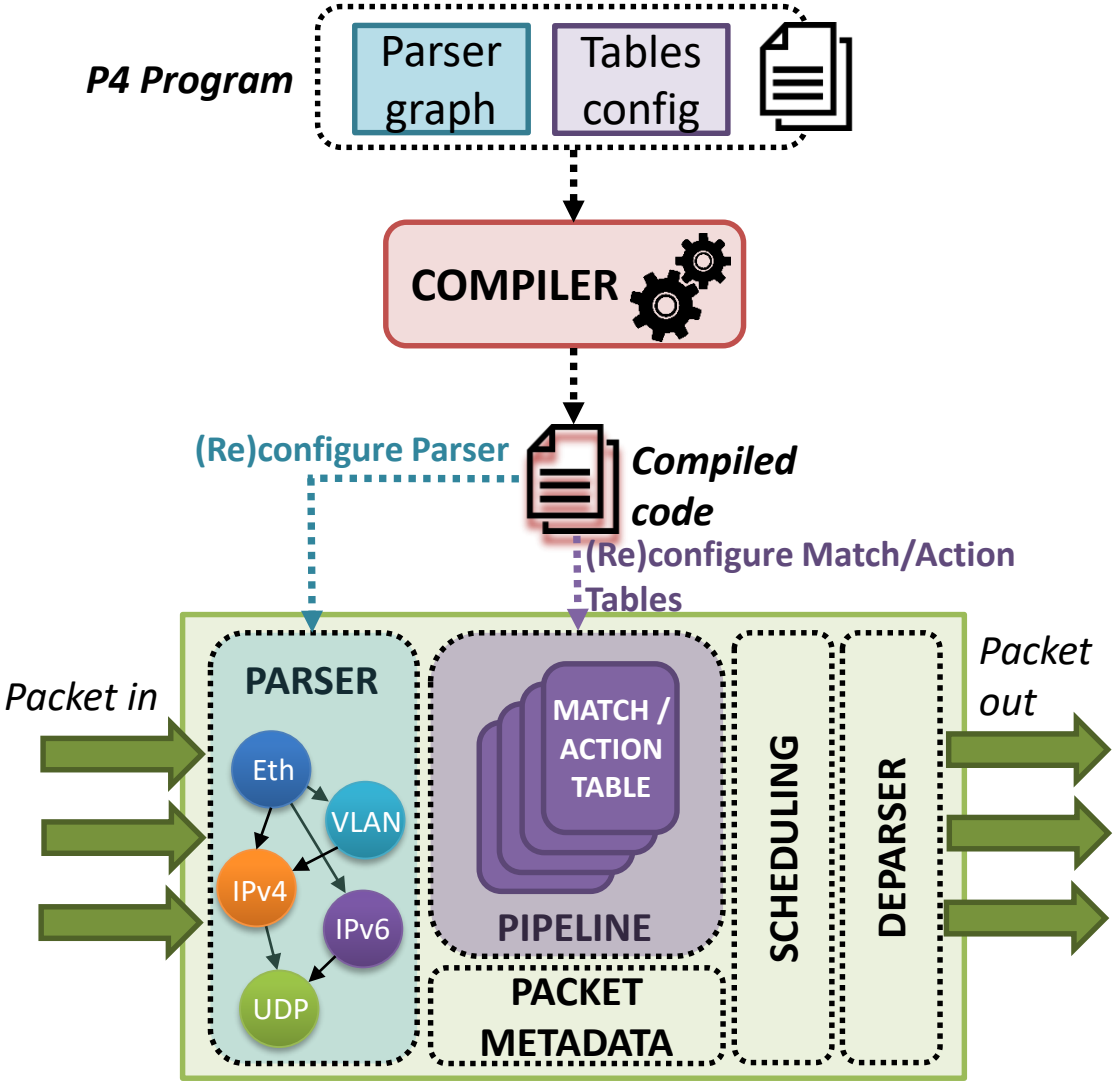
[1] http://blog.sinefa.com/blog/2015/08/13/9-limitations-to-be-aware-of-when-considering-netflow-for-visibility

Programmable data planes



- Bring advancements in the **Software-Defined Networking** paradigm
  - Higher degree of **flexibility** (e.g. add/remove support to new/unused protocols and functionalities)
  - More **efficient usage of resources** (e.g. memory, CPU)
  - **Software-like design** and development
- Many different solutions and scenarios, we explore two:
  - In-network: **P4 architecture** (for switches)
  - Edge: extended Berkeley Packet Filtering (**eBPF**, for Linux-based end-host devices)

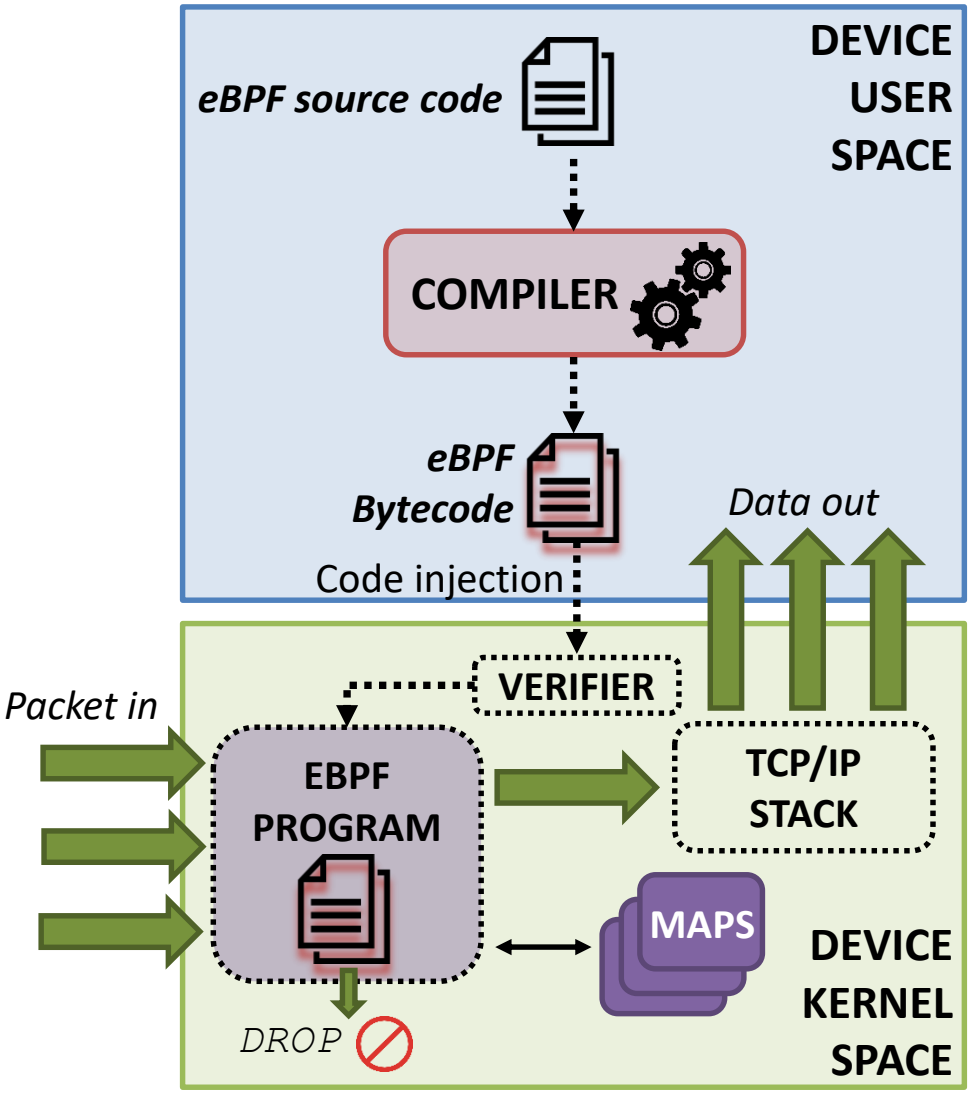
P4 switch high-level architecture



- **PISA** (Protocol Independent Switch Architecture)
  - Packet parsed into headers
  - Headers, intermediate results and metadata can be used for matching and actions
  - Headers can be modified, added or removed
- Very suitable for network flow monitoring
  - In-band Network Telemetry [2]
  - Sketching algorithms in data plane [3]

[2] In-band Network Telemetry (INT), https://p4.org/assets/INT-current-spec.pdf
[3] F. Pereira et al., "Secure network monitoring using programmable data planes," 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, 2017, pp. 286-291.

Extended Berkeley Packet Filter (eBPF)



- **eBPF**
  - Linux kernel technology
  - Special-purpose virtual environment running eBPF programs
- **eBPF programs**
  - Reside in the Linux kernel
  - Generated by user-space applications and injected via system calls
- **Notable functionalities**
  - **Access** incoming/outgoing packets and collect information
  - **Filter** incoming/outgoing packets
- Very suitable for efficient network flow monitoring [4], also for anomaly detection [5]

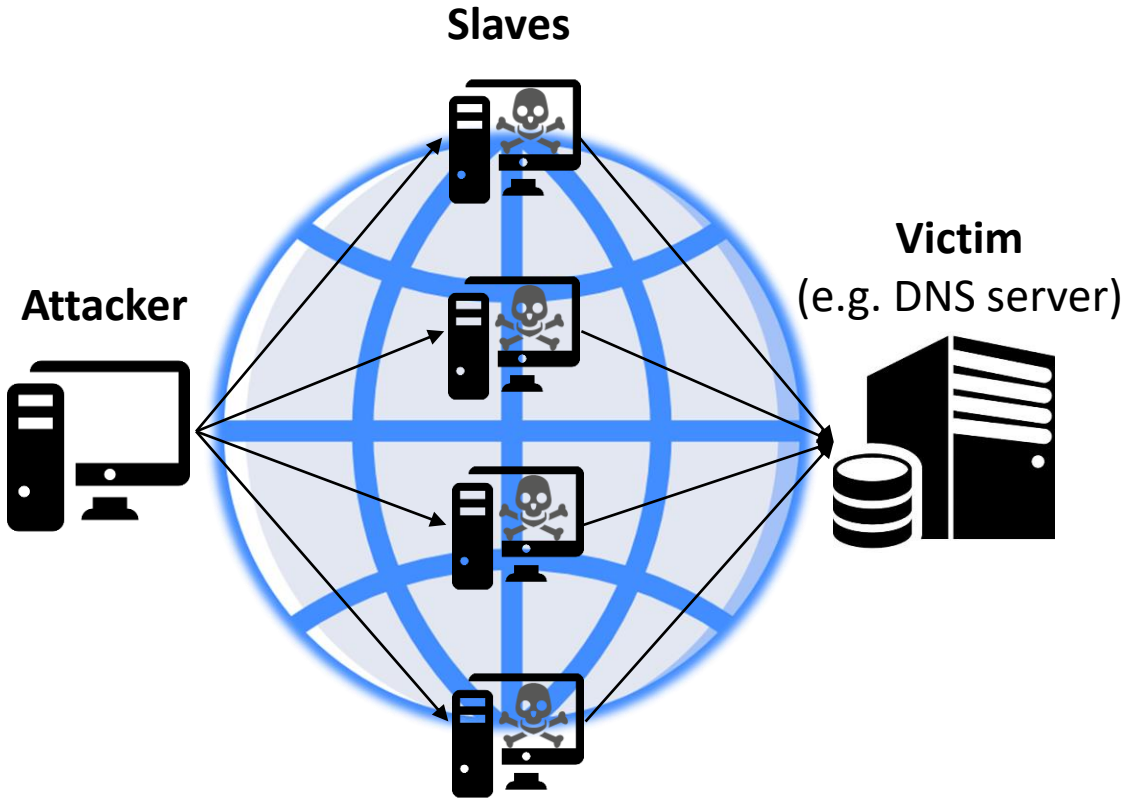
[4] Recap: High-performance Linux Monitoring with eBPF, https://www.weave.works/blog/recap-high-performance-linux-monitoring-with-ebpf/
[5] G. Bertin, "XDP in practice: integrating XDP into our DDoS mitigation pipeline", https://netdevconf.org/2.1/papers/Gilberto_Bertin_XDP_in_practice.pdf

P4 vs. eBPF: a comparison

	P4	eBPF
Program functionalities / features		
Programming language	Domain-specific	C
Memory allocation	Static	Static
Maximum stack space	No limit	512 bytes
Maximum number of instructions per program	No limit	4096
Loops/recursive functions	✗	✗
Pointers/references	✗	✓
Global variables	✓	✗
Wildcarding mechanim for table lookup	✓	✗
Additional network functionalities		
Deep-packet inspection	✗	✓
Packet fragmentation	✗	✗
New packet generation (e.g. ICMP reply)	✗	✗
Processing of packet trailers	✗	✓
Other aspects		
Hardware	Support PISA / FPGA	Generic-purpose
Best applicability scenario	In-network	Edge

Use case: DDoS detection/mitigation

- We present a use case for the **detection/mitigation of DDoS attacks**
- **Two scenarios**: exploiting network flow monitoring capabilities provided by
  1. P4-enabled network
  2. eBPF at the edge



Scenario 1
DDoS detection/mitigation in a P4-enabled network

- **P4-enabled** SDN network

SWITCH DATA PLANE

PACKET METADATA

PARSER

PIPELINE

COMPARISON WITH LAST COUNT-MIN SKETCH

SUSPICIOUS BUCKETS (IF ANY)

B

SKETCH

A

FLOW COUNTERS (COUNT-MIN SKETCH)

HASHED BUCKETS

SRC IP BLACKLIST (BLOOM FILTER)

SRC IP

$h_1()$ $h_2()$ $h_3()$

0 1 1 0 1

DST IP

$h_A()$ $h_B()$ $h_C()$ $h_D()$

DOES DST IP HASH TO SUSPICIOUS BUCKETS?

NO

PASS

YES

ADD SRC IP TO BLACKLIST

SCHEDULING

DEPARSER

Packet in

Packet out

IP HEADER

PASS

PASS

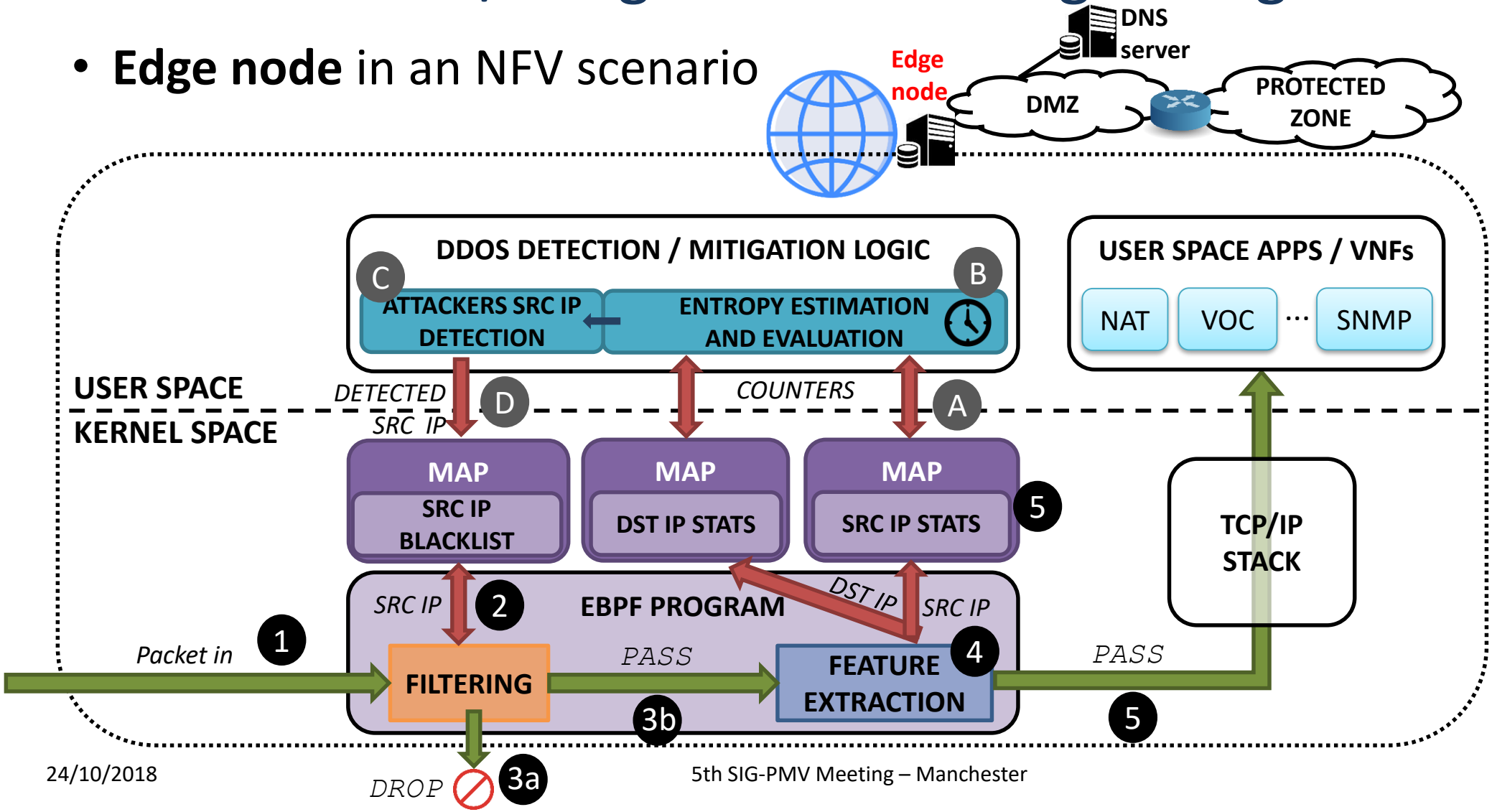
DROP

DROP

1 2 3a 3b 4 5a 5b 6

Scenario 2
DDoS detection/mitigation at the edge using eBPF

- **Edge node** in an NFV scenario



Next steps (1)

1. (P4/eBPF) **Evaluation** of DDoS strategies in terms of *detection accuracy* and *detection time*
   - Comparison with existing solutions (e.g. based on NetFlow)
2. (P4) **Setup of a testbed** composed of three *Wedge100BF-32X* programmable switches, each equipped with
   1. Thirty-two 100G QSFP28
   2. Barefoot Tofino 3.2 Tbps chip
3. (P4/eBPF) Is it possible to define an improved **«generic» algorithm** using the subset of functionalities offered by both technologies?
   - Right now, two different algorithms for DDoS detection/mitigation
   - Are sketches implementable in eBPF?

Next steps (2)

4. (P4) Definition of a strategy for effective **partial deployment** of programmable switches
   - Where to deploy a limited number of programmable switches in the network?
   - Objective: minimizing the DDoS detection/mitigation performance degradation




Thank you for your kind attention!

FONDAZIONE BRUNO KESSLER | CREATE-NET