

Packet Processing Programming on SmartNICs

ACM SIGCOMM 2026

Programming SmartNICs: From Packet Processing to Programmable Transport

Muhammad Shahbaz

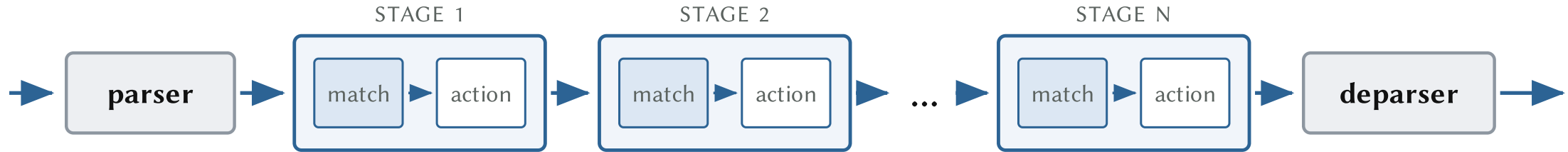
University of Michigan

What does it mean to program packets on a SmartNIC?

- What **is** a packet-processing program?
- How is one **written**? — P4, DPL, DOCA Flow
- What **changes** when it moves into a NIC?

The exercise at 10:45 uses every construct introduced here.

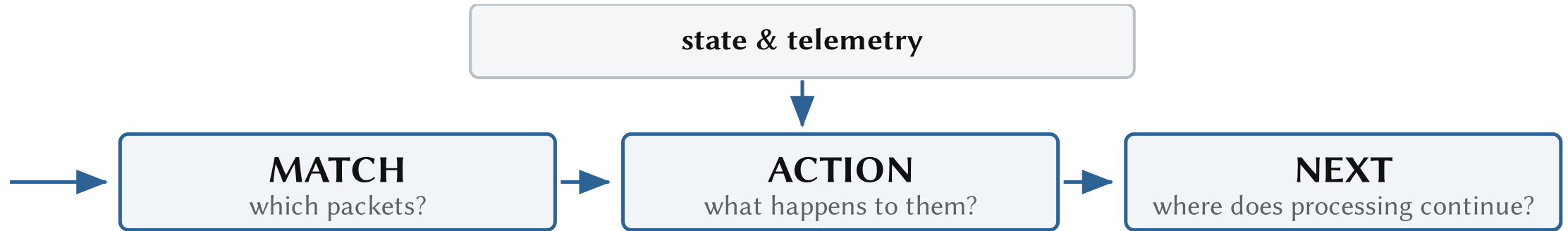
The data-plane pipeline



Reconfigurable Match Tables — Bosshart et al., SIGCOMM 2013. Programmed in P4.

The abstraction is unchanged, but the location is moved.

The mental model



P4, DPL and DOCA Flow are three notations for these three questions.

Match: four kinds of lookup

Lookup	Used for
Exact match	a flow, a connection, a MAC, a VNI
Longest prefix	IP routing
Ternary / masked	ACLs, classification
Range	port ranges, policy

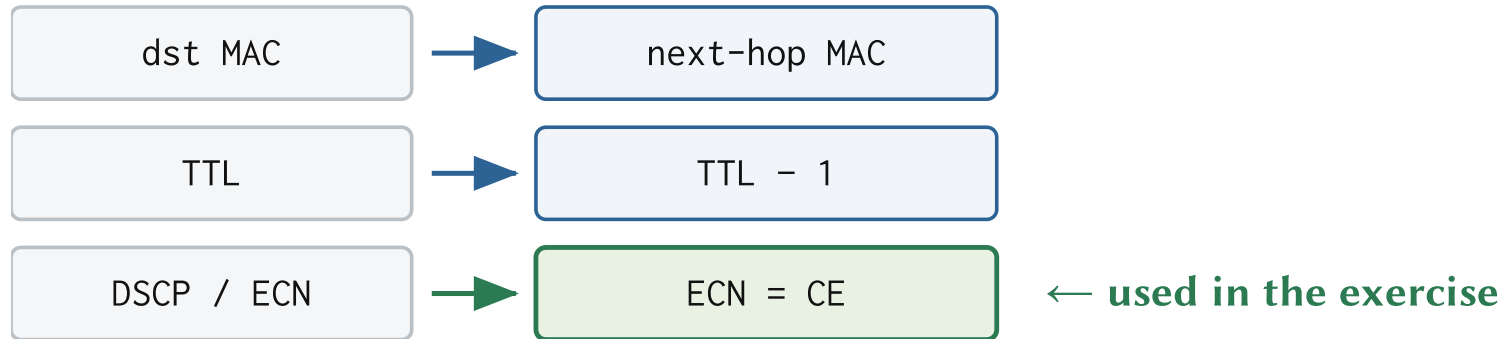
The exercise uses **exact match** and the NIC's **random field**.

Every lookup is a hardware budget

- **Key width** — bits matched per lookup
- **Entry count** — how much policy fits
- **Memory type** — SRAM-like is cheap, TCAM-like is not
- **Stage depth** — chained lookups consume the pipeline

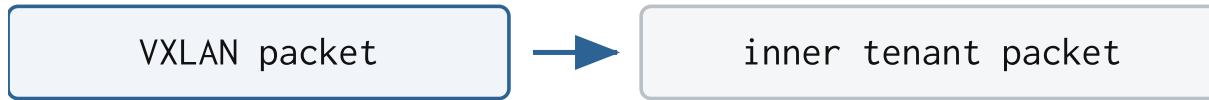
A range match expands into multiple ternary entries. The cost is not visible in the source.

Action: rewrite a field



Two bits, rewritten in hardware, at line rate.

Action: encapsulate, decapsulate



Together, these two are what makes network virtualisation work.

Action: tag and steer



The wire format is unchanged. The tag is carried between pipes, then discarded.

A table is a template. An entry is a value.

THE TABLE — DECLARED ONCE

key: ipv4.dst_addr
actions: forward, drop
which field is looked up, and what may be done



THE ENTRIES — INSTALLED AT RUN TIME

10.0.0.0/8 → forward(3)
10.1.0.0/16 → forward(7)
0.0.0.0/0 → drop

Fixed when the pipeline is built: the fields, the action set, the memory it needs.

Changeable while traffic flows: the entries.

The table names the **fields**. The entries carry the **values**.

The same split, in all three interfaces

	Structure	Runtime state
P4	tables and actions	entries, via P4Runtime
DPL	tables and actions	entries, via P4Runtime
DOCA Flow	pipes, built by the application	pipe entries

Building the pipeline defines the **structure**.
Installing entries defines the **policy**.

Who fills the tables?

1 · Provisioned

A controller computes the state and installs it.

Routing · ACLs

2 · Reactive

The first packet goes to software; the rest are offloaded.

Flow offload · conntrack

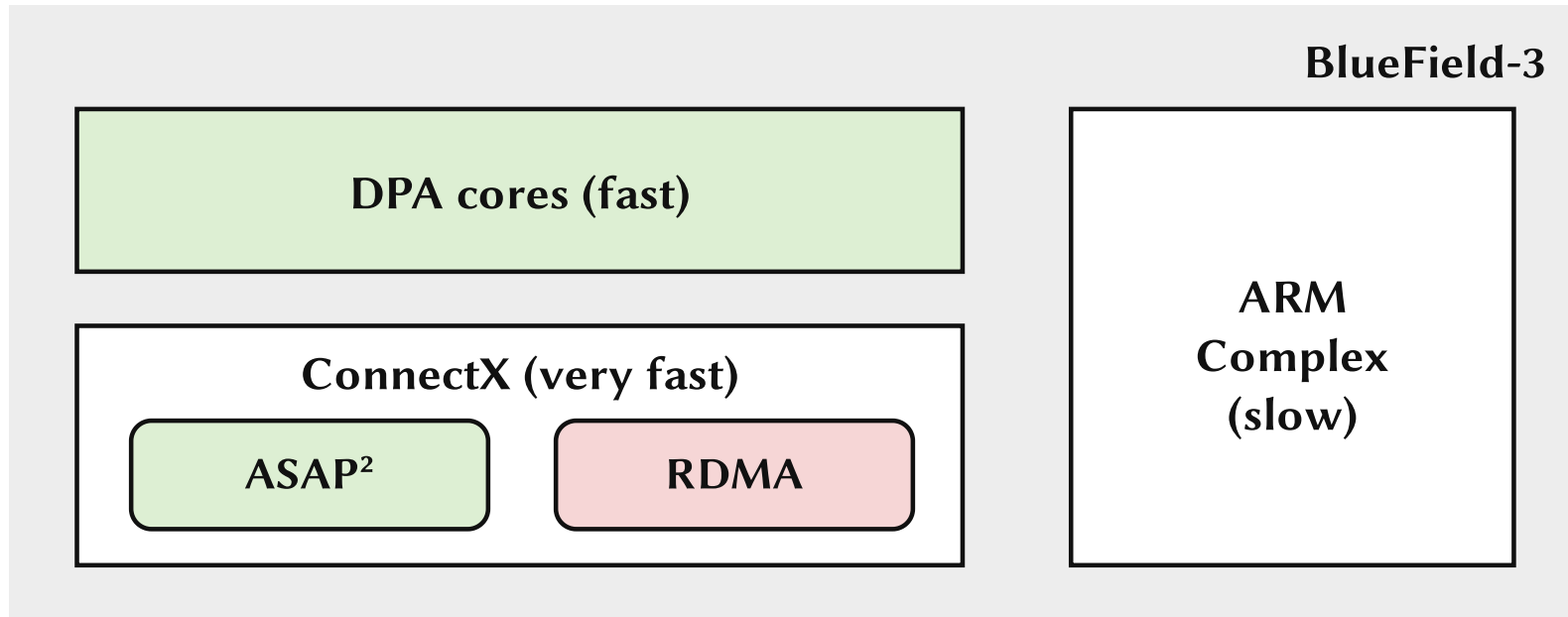
3 · Autonomous

The pipeline inserts its own entries, at line rate.

Learning · aging



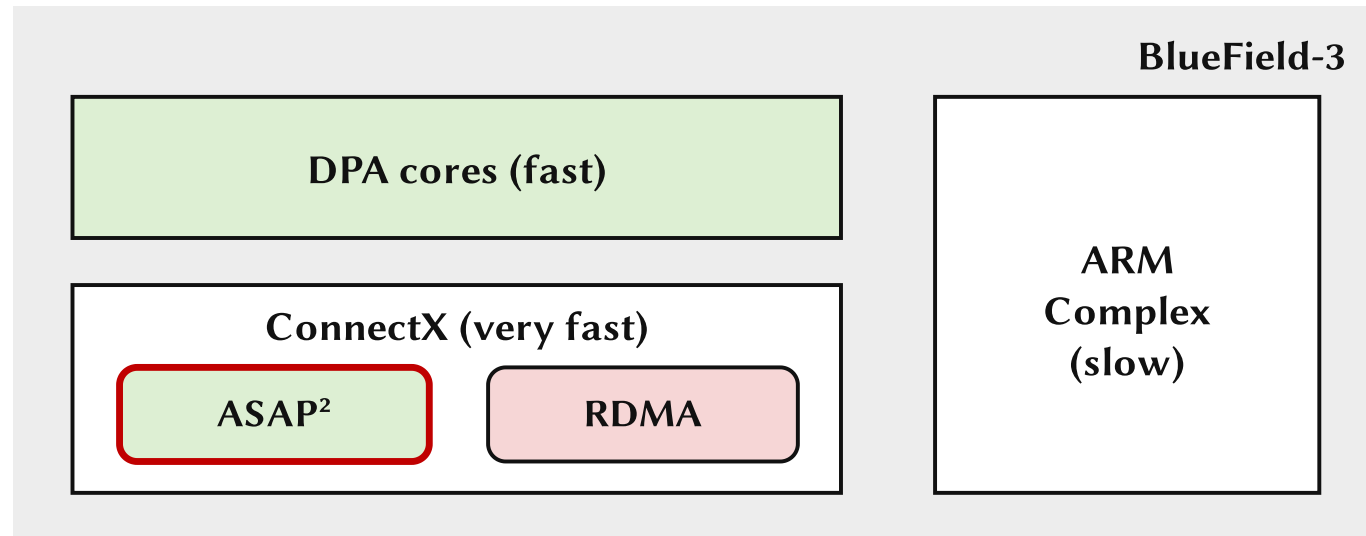
The same model, but inside a NIC



Three programmable domains, at three different speeds.

ASAP² — Accelerated Switching and Packet Processing. DPA — Data-Path Accelerator.

The same model, but inside a NIC



- **ASAP²** — the packet pipeline. *The subject of this section.*
- **DPA cores** — per-event, stateful. Covered in the next section.
- **Arm complex** — runs the control program, then idles.

The NIC pipeline is not a switch pipeline

Switch — RMT

- A fixed sequence of physical stages
- Headers re-serialised by a **deparser**
- The architecture **P4 was designed for**

NIC — run to completion

- Processing completes per packet
- **No deparser stage** — modification and reparsing are in line
- Accelerators invoked as primitives

The same abstraction, a different execution model.

“While P4 semantics imply a staged pipeline based on a feed-forward RMT architecture, NVIDIA's DPU architecture follows a run-to-completion dRMT model.”

— NVIDIA, DOCA Pipeline Language Services Guide

Three interfaces, one computation

P4

A language.
Portable model.

Compiled.

DPL

A language.
This NIC.

Compiled.

DOCA Flow

A C library.
The application.

Built at run time.

P4 and **DPL** are compiled languages. **DOCA Flow** is a library the application calls.

The behaviour in P4

```
action mark_ce() {  
    hdr.ipv4.ecn = 3;  
}  
  
table ipv4_policy {  
    key      = { hdr.ipv4.dstAddr : lpm; }  
    actions = { mark_ce; forward; drop; }  
}
```

P4

- The program fixes the **key** and the **action set**
- A control plane supplies the **entries**
- No vendor appears in the program

The behaviour in DPL

```
action mark_ce_and_fwd(nv_logical_port_t port) {  
    nv_set_ip_ecn(headers, 2w3);  
    ce_ctr.count();  
    nv_send_to_port(port);  
}
```

```
table ipv4_policy {  
    key      = { headers.ipv4.dst_addr : exact; }  
    actions = { mark_ce_and_fwd; NoAction; }  
    direct_counter = ce_ctr;  
}
```

- **Keeps** P4's grammar, and P4Runtime
- **Drops** registers, digests, the deparser
- **Adds** the ASIC's own primitives as externs

DPL is not a new syntax. It is P4 mapped onto this hardware.

DOCA Flow: the anatomy of a pipe



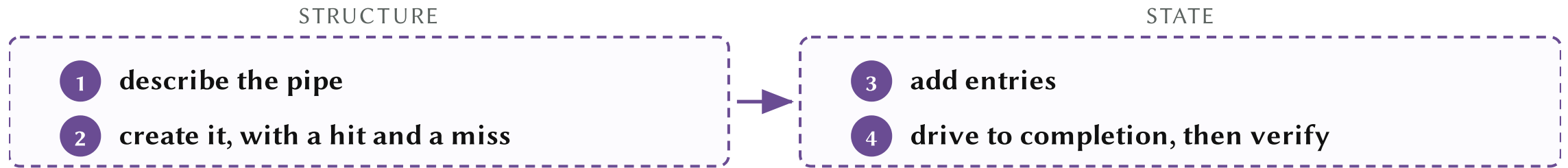
A pipe is a flow table rather than a pipeline stage. Monitoring sits beside the match.

DOCA Flow: pipe types

Type	What it does
BASIC	An ordinary match-action table
CONTROL	Entries carry priorities, resolving overlapping matches
LPM	Longest-prefix match, for routing-shaped lookups
ACL	Five-tuple matching with masks
ORDERED_LIST	A fixed sequence of actions per entry
HASH	Selects an entry by computed index rather than by matching

The exercise uses **BASIC**.

DOCA Flow: constructing a pipe



In DOCA Flow, `0xff` in a *pipe* marks a field as participating. The value itself appears only in an *entry*.

The behaviour in DOCA Flow

```
/* the pipe: which field participates */
actions.outer.ip4.dscp_ecn    = 0xff;
doca_flow_pipe_create(cfg, &hit, &miss, &pipe);

/* the entry: the value it writes */
e_actions.outer.ip4.dscp_ecn = 0x03;
doca_flow_pipe_add_entry(q, pipe, ...);
doca_flow_entries_process(...);
```

A language **describes** a pipeline.

A runtime API **controls** it.

The pipeline frequently belongs to an application that already exists.

One behaviour, three interfaces

P4

```
hdr.ipv4.ecn = 3;
```

A field assignment.

DPL

```
nv_set_ip_ecn(  
    headers, 2w3);
```

An extern implemented by the ASIC.

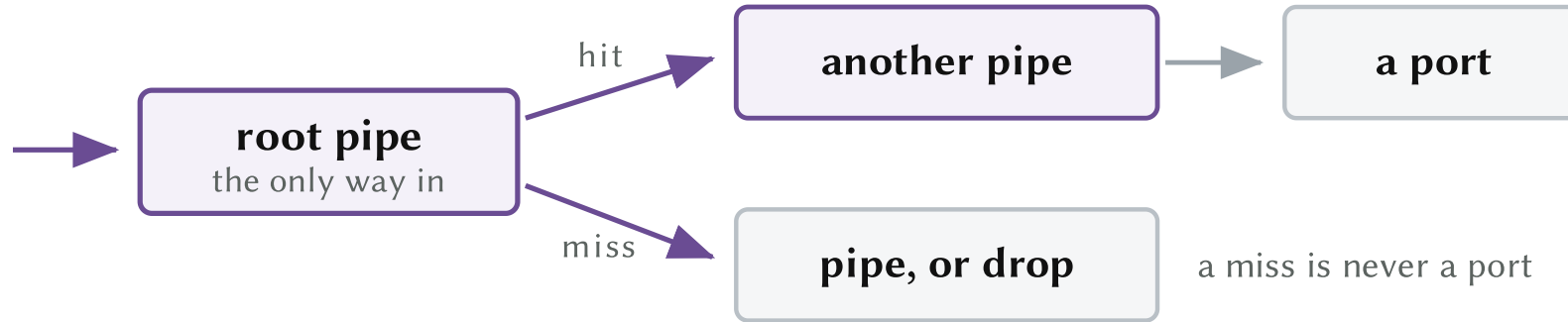
DOCA FLOW

```
e_actions.outer  
    .ip4.dscp_ecn = 0x03;
```

A value on an entry.

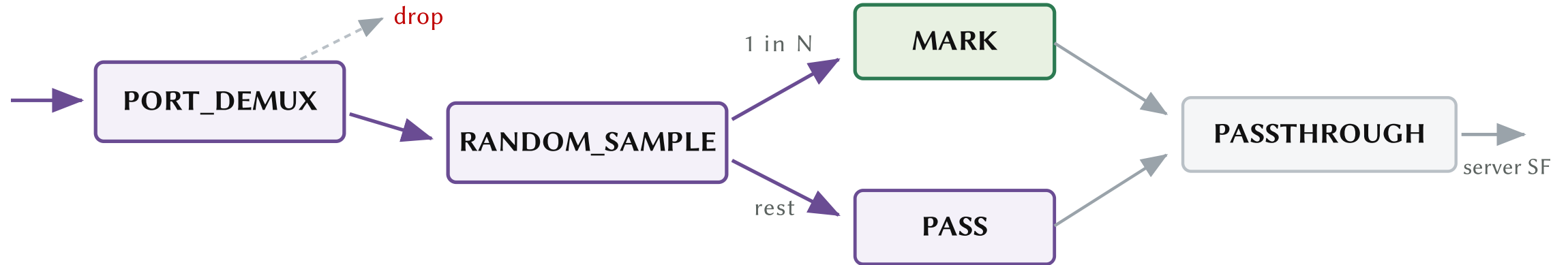
One intent, three interfaces, three lifecycles.

A pipeline is a graph, not a script



The provided forwarder sends wire traffic to a PORT. Sending it to a PIPE instead is the entire difference between a forwarder and a pipeline.

After the break: Mark ECN

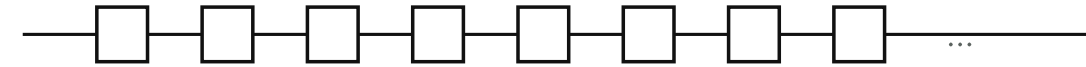


Pipe	Function	
PORT_DEMUX	create_root_pipe	TODO 1
MARK and PASS	create_forward_to_sf_pipe	TODO 2
RANDOM_SAMPLE	create_sampling_pipe	TODO 3
PASSTHROUGH	create_passthrough_pipe	provided

- One function builds **both** forwarding pipes — it is called twice, once with marking off
- The return path from the server is forwarded **unmarked**
- --percent N sets the marked fraction

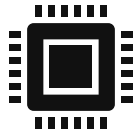
GigaFlow: a smart cache for a SmartNIC

PACKET PROCESSING PIPELINE



a chain of match-action tables

Megaflow



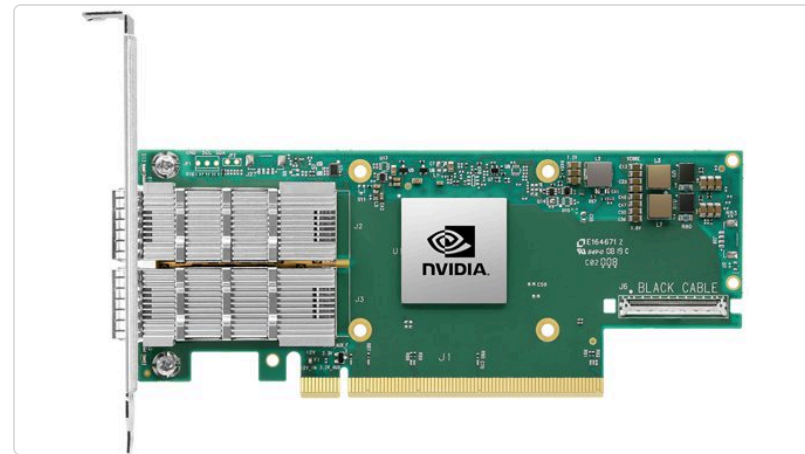
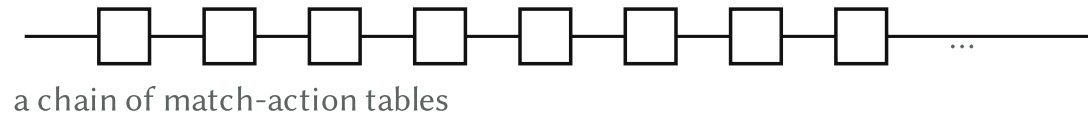
It assumes

- Multi-table lookups are **expensive**
- Pipelines are **fixed**

Both hold on a CPU.

GigaFlow: a smart cache for a SmartNIC

PACKET PROCESSING PIPELINE



It assumes

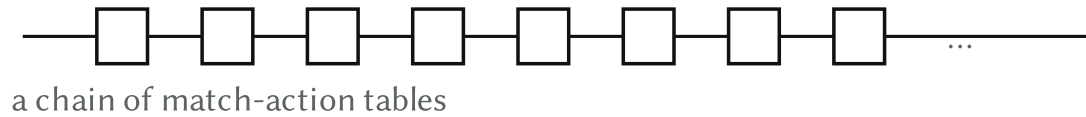
- Multi-table lookups are expensive
- Pipelines are fixed



On a SmartNIC **neither holds**, and the pipeline itself is programmable.

GigaFlow: a smart cache for a SmartNIC

PACKET PROCESSING PIPELINE



Caches are distributed *across* the pipeline rather than placed in front of it.

450× more rule coverage

50× higher hit rate

Moving a function onto the NIC changes what the right design is.

Where match-action runs out

Good at

- Classification and lookups
- Header rewrite, encap, decap
- Counters and meters
- **Bounded** state

Awkward for

- Per-flow algorithms
- Timers and events
- Large dynamic state
- **Messages**, not packets

Transport requires precisely what this model handles least well.

Hence the **DPA** — the Data-Path Accelerator, the card's second programmable engine.

**Packet processing takes the packet
as the unit of computation.**

Transport maintains state across packets and across time.

Next: **Mina Tahmasbi Arashloo** — programmable transport.