Smashing OpenFlow’s “atomic” actions: programmable data plane packet manipulation in hardware

Salvatore Pontarelli, Marco Bonola, Giuseppe Bianchi
CNIT / University of Roma Tor Vergata,
Email: {name.surname}@uniroma2.it

cnit

Beba
BEhavioural BAsed forwarding

Outline

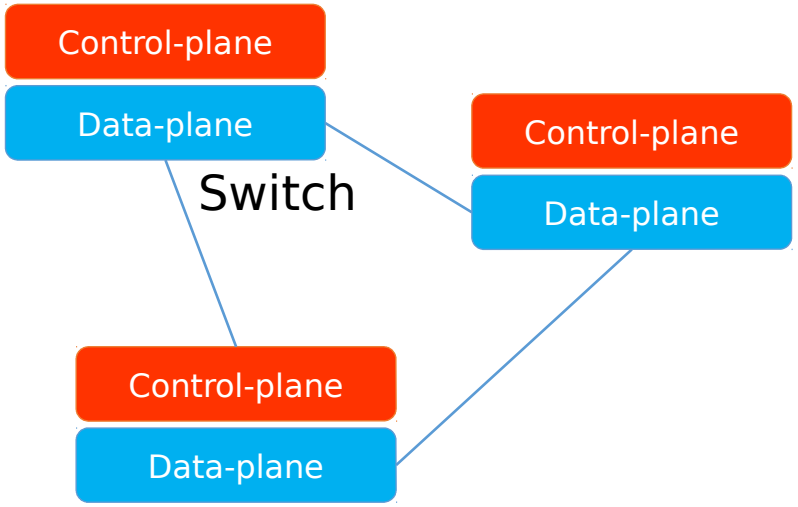
- Introduction
- Programmable SDN action
- PMP (Packet Manipulation Processor) Architecture
- Example of PMP applications
- Conclusions

cnit

Beba
BEhavioural BAsed forwarding

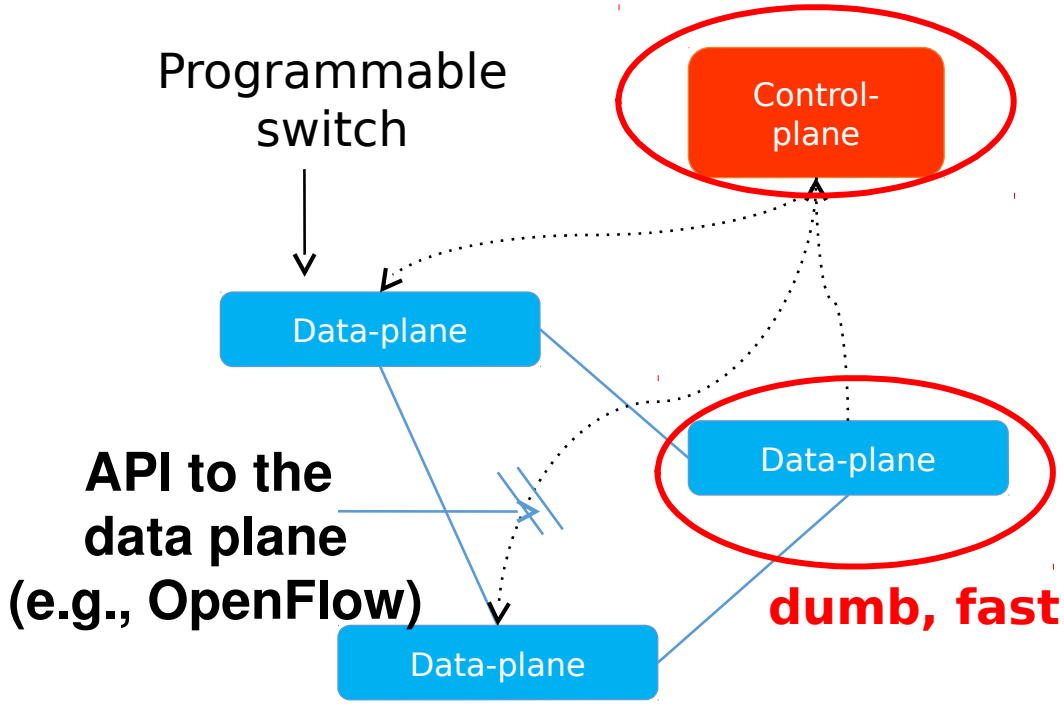
The SDN paradigm

Traditional networking



Software-Defined Networking

smart, slow, (logically) centralized

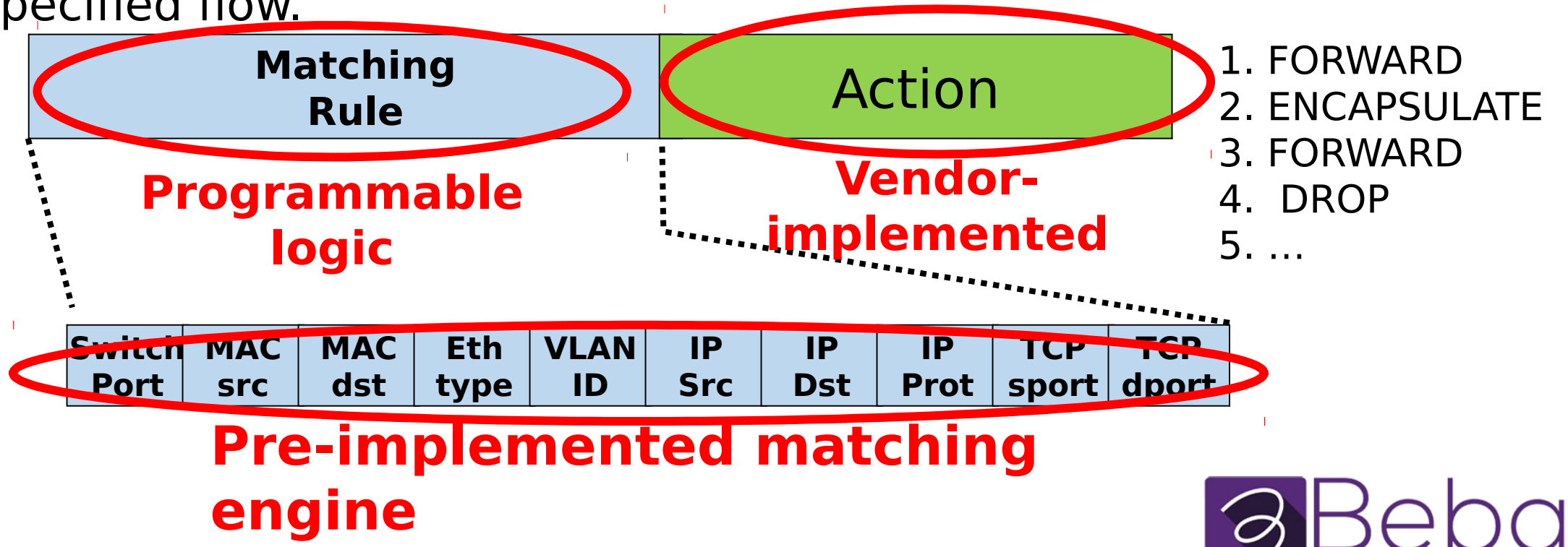


cnit

Beba
BEhavioural BAsed forwarding

OpenFlow match/action abstraction

OpenFlow proposes an abstract model of a programmable flow table which permits the device programmer to broadly specify a flow via an header matching rule, associate forwarding/processing actions to the matching packets, and access statistics associated to the specified flow.



cnit

Beba
BEhavioural BAsed forwarding

Action Programmability

- Actions in OpenFlow have mainly remained “atomic” and to the best of our knowledge very little work has addressed their flexibility.
- The programmer can only “select” which action should be associated to the outcome of a match, being such selection restricted to a set of actions (e.g., drop, output to port, push/pop VLAN/MPLS tag, etc) **preimplemented** in the device by the vendor.

Action programmability can:

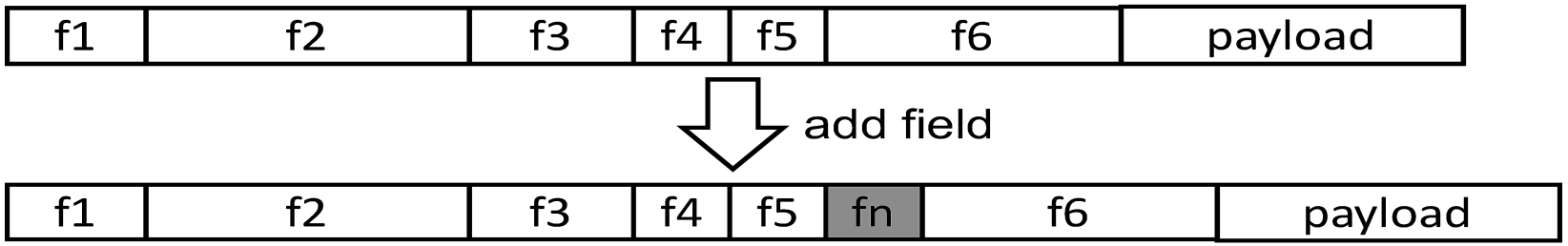
- Allowing SDN switches to run complex network task such as NAT, generic encapsulation etc.
- Help the introduction/deployment of of new/enhanced network protocols (new encapsulation methods, new/experimental protocol fields, etc.)
- Adding new functionality such as programmable packet generation or in-band network measurements and debug.

SDN action classes:

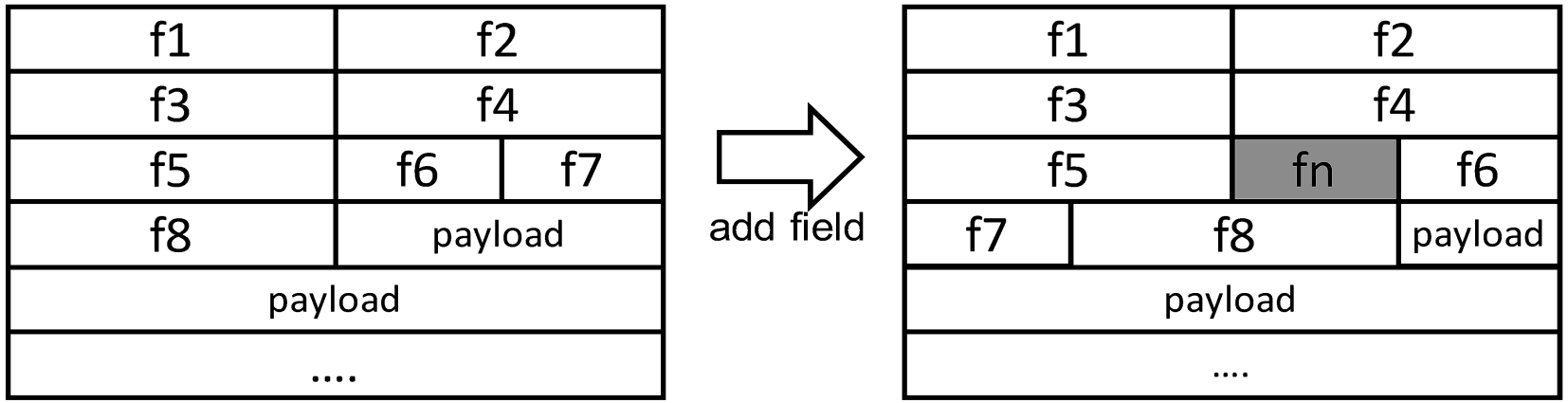
We identify three classes of actions starting from the set of pre-implemented actions provided by OpenFlow (and also by the newcomers SDN languages such as P4):

1. Header fields actions: these actions will add/modify/remove one of the header fields of the packet that is processed
2. Packet level actions: these actions acts at packet level performing drop(), clone(), recirculate() actions
3. Packet generation actions: while not directly available in current SDN, we believe that this will be a feature of future SDN platforms.

Header fields actions



(a) the packet before and after the add field operation



the packet stored in a 32 bits memory before and after the add field operation

cnit

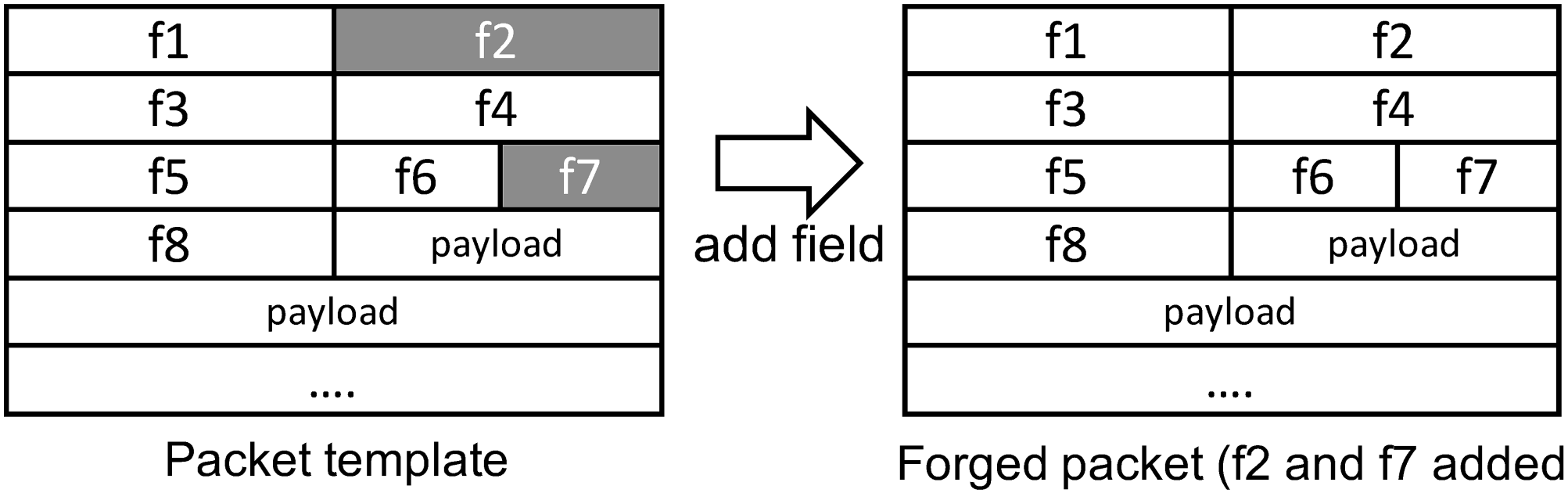
Beba
BEhavioural BAsed forwarding

Dispatching actions

These operations are a set of read/write from/to the queue memory in which the packet is stored, to another location (an I/O memory mapped location (an output port) or another location inside the buffer memory (e.g. to implement the P4 clone() operation).

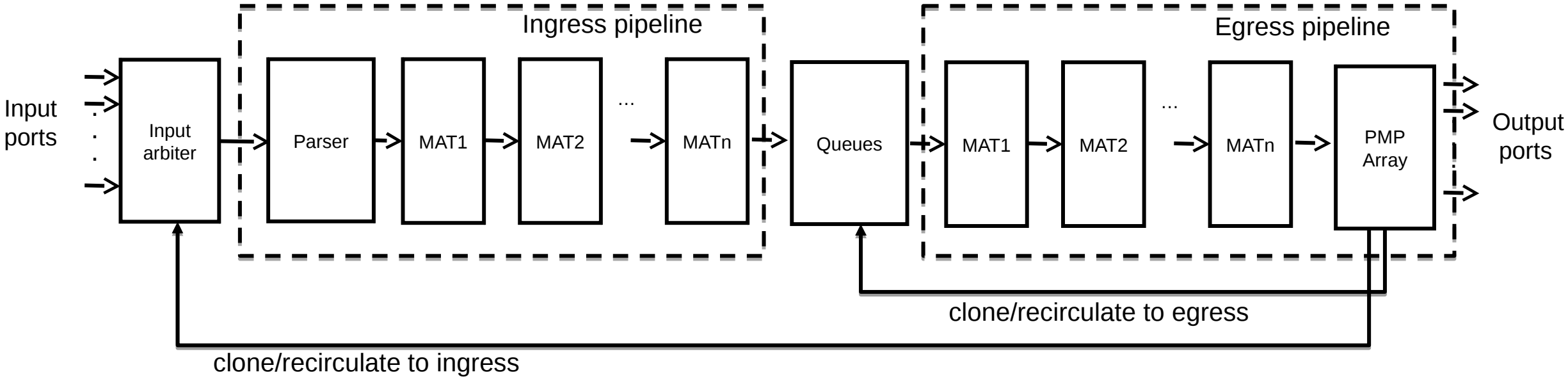
- the bandwidth of the memory read/write operation is directly related to the memory data-width
- the use of specific instructions that move data from one memory address to another memory address can increase the throughput.
- Memory can be shared between switch queue and PMP

Packet generation actions



The actual packet to send outside the switch is forged copying the data coming from the template, with suitable modification of some packet headers/data.

Switch pipeline architecture



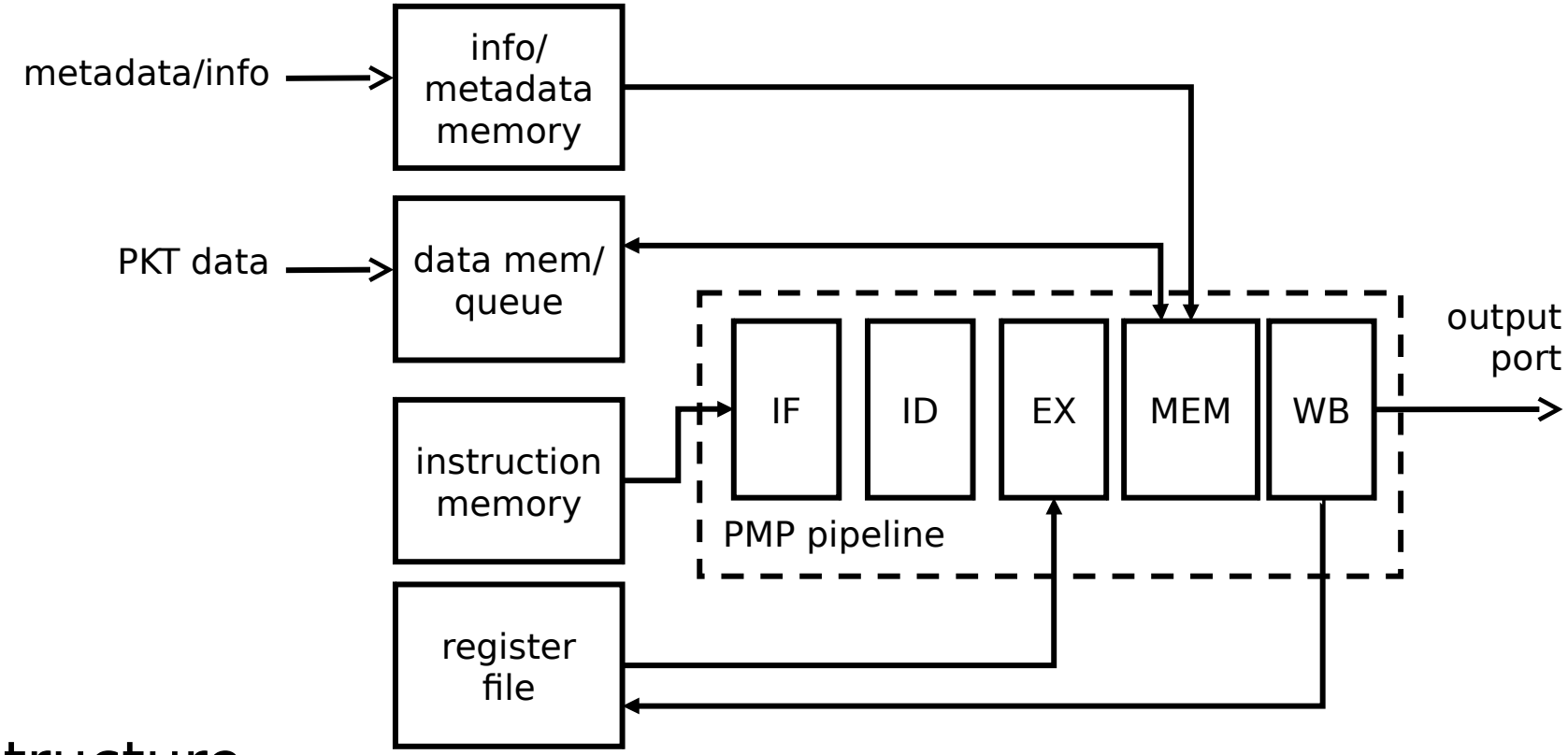
The current SDN architectures are based on a pipeline of MAT (match-action table) stages forming an ingress and an egress pipeline.

The PMP can be inserted at the end of the egress pipeline (it is the mirror block of the ingress parser)

cnit

Beba
BEhavioural BAsed forwarding

PMP architecture



- RISC-like structure
- plain memory structure (no cache)
- multiple memory accesses

cnit

Beba
BEhavioural BAsed forwarding

PMP instruction set

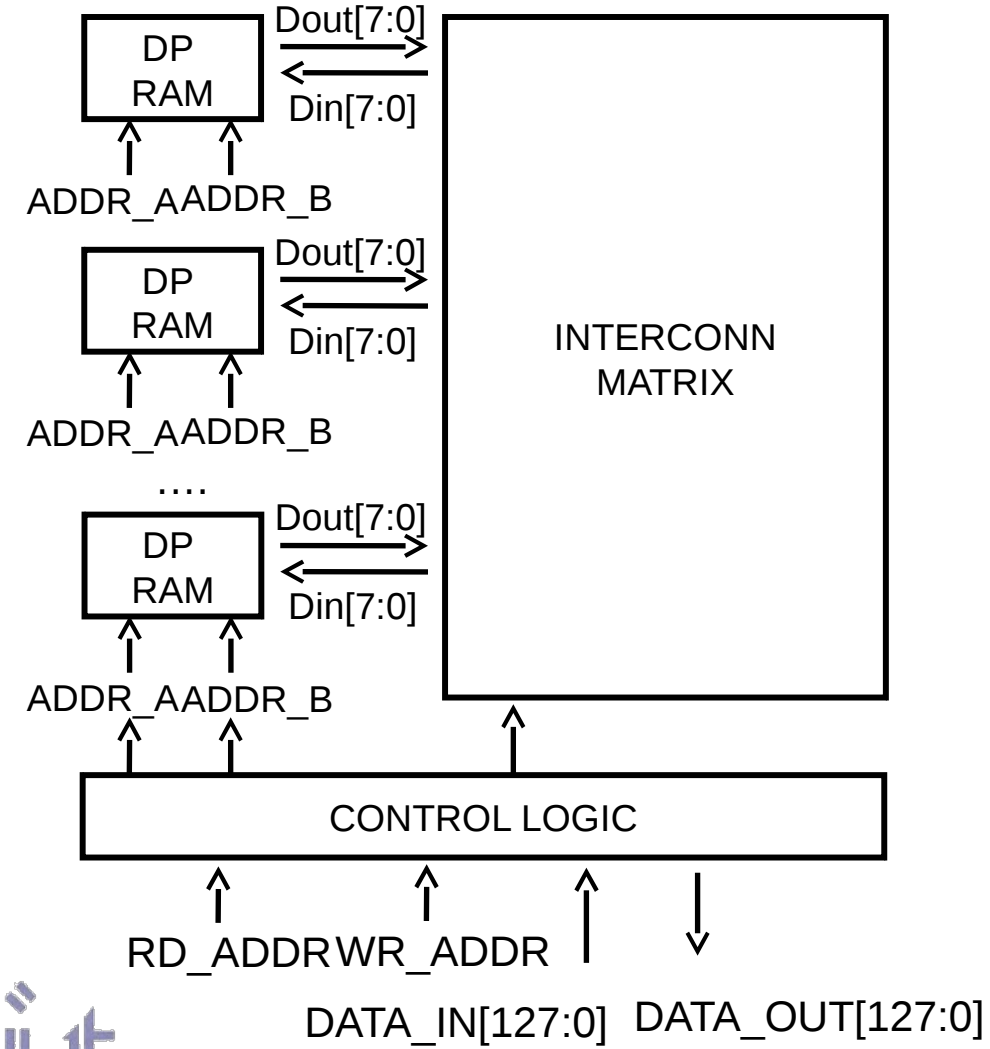
Instruction Type	Instructions	Memory mode	Operands	Note
Logic ALU Operations	NOP, AND, OR, XOR, XNOR,NOT	register-register	rd,rs1,rs2 rd,rs1,imm	standard logic operations
Arithmetic ALU Operations	ADD,ADC, SUB,SBC,MUL,	register-register	rd,rs1,rs2 rd,rs1,imm	standard arithmetic operations
Shift/Rotate Operations	LSL (Logical Shift Left) LSR (Logical Shift Right) ASR (Arithmetic Shift Right) ROR (Rotate Right)	register-register	rd,rs1,rs2 rd,rs1,imm	performs logic and arithmetic shift/rotate operations
Control flow	B(cond) (Branch with condition) BL(Branch and Link with condition) RET (Return), HALT	register-register imm	rs1	The conditions check the status flags and if matched executes the branch. BL also save the pc value in the link register (r14) HALT ends the PMP execution.
Load/Store	ldb, ldh,ldw stb, sth,stw	register-memory memory-register	rd, imm	the operations move 8,16 or 32 bits
mov	movb,movh,movw,movd,movq	memory-memory	rd,addr	movs move up to 128 bits from [addr] to [rd]
out	outb,outh,outw,outd,outq	memory-port	rd, addr	outs move from [addr] to the rd output port.
memory data movement	meml (Memory loop)	memory-memory	rd,rs1,rs2	movl moves rs2 bytes from [rs1] to [rd]
output data movement	outl (output loop)	memory-port	rd,rs1,rs2	moves rs2 bytes from [rs1] to the rd output port

- minimal ISA
- loop instructions (DSP-like)
- directly I/O mov instructions
- multiple size instructions (B/H/W/D)

cnit

Beba
BEhavioural BAsed forwarding

unaligned PMP memory



- combined PMP memory / packet queue
- multiple 8 bits banks for unaligned access
- dual port for PMP/switch interface

cnit

Beba
BEhavioural BAsed forwarding

Developed applications

- Network Address and Port Translation

  The throughput achievable goes from 11.6 Gb/s (worst case) to 90 Gb/s (max size packets)

- ARP reply generation

  The ARPreply code is always executed in 18 clock cycles, which correspond to a throughput for this application of around 28.4 Gb/s.

- IPinIP encapsulation

  (harder than other encapsulation types: TTL, IP fragmentation etc). The throughput achievable goes from 12.2 Gb/s (worst case) to 90 Gb/s (max size packets)

Conclusions

- We proposed an architecture to adding action programmability to SDN platforms
- We selected a specific instruction set tailored for packet manipulation
- we designed a CPU achitecture for packet manipulation able to provide multiple 10 Gb/s of troughput.
- We tested the proposed CPU with a several network application to test the effectiveness of the PMP

cnit

Beba
BEhavioural BAsed forwarding