AUGUST 2018

Programmable Data Plane at Terabit Speeds

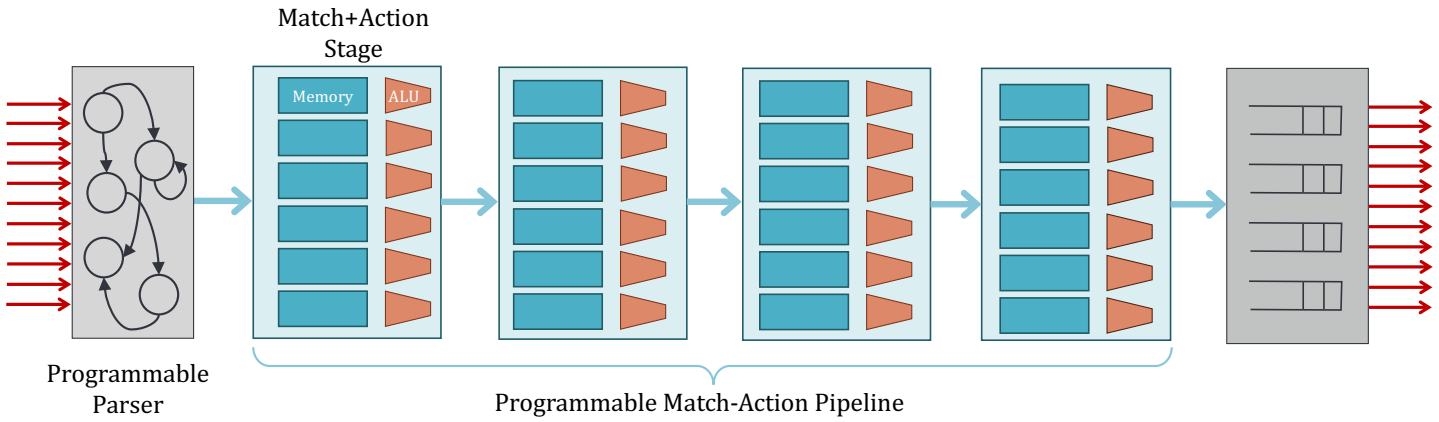
Milad Sharif
SOFTWARE ENGINEER

BAREFOOT

PISA: Protocol Independent Switch Architecture

BAREFOOT

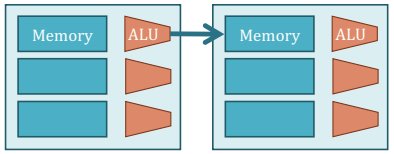
PISA – Block Diagram



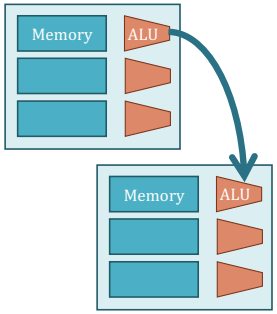
BAREFOOT

PISA – Match-Action Unit

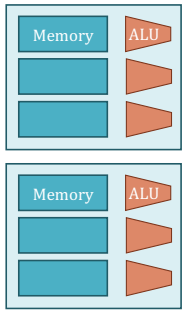
- Multiple simultaneous lookups and actions can be supported



Sequential Execution
(Match dependency)



Staggered Execution
(Action dependency)



Parallel Execution
(No dependency)

Introducing Tofino

BAREFOOT

6.5Tb/s Tofino™ – Overview

Four Match+Action Pipelines

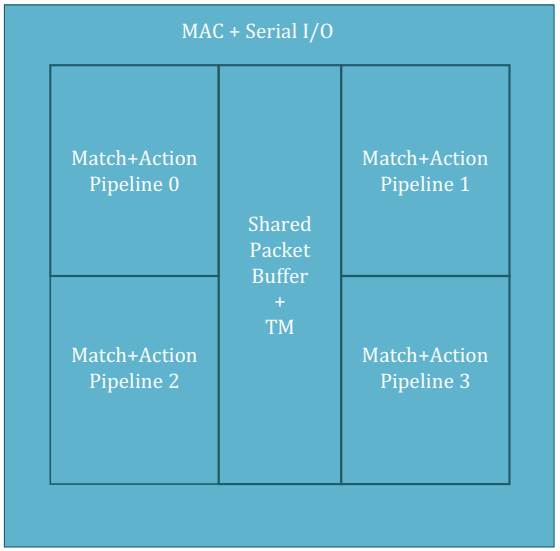
- Fully programmable PISA Embodiment
- Single Shared Packet Buffer
- 16nm technology

Port Configurations

- 260 × 25 SerDes, support 1G, 10G, 25G, 40G, 50G, and 100G ports

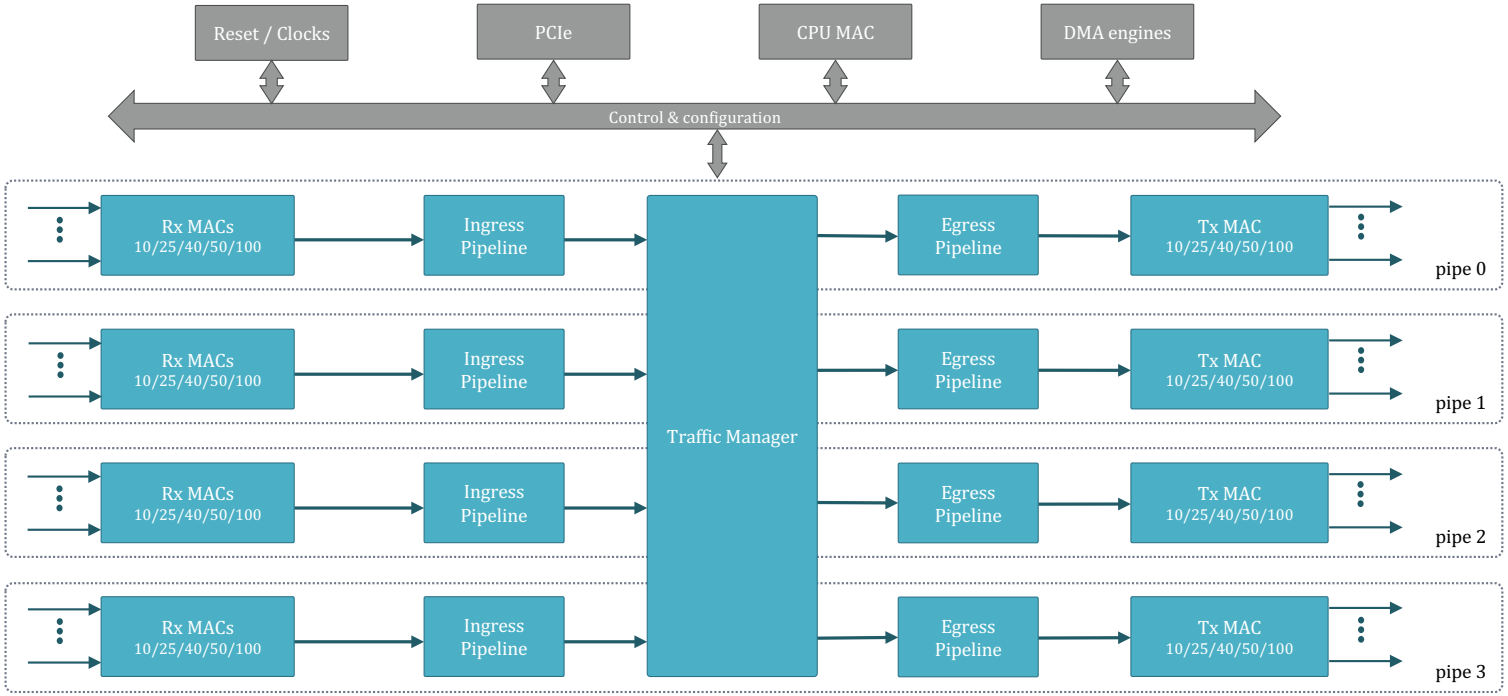
CPU Interfaces

- PCIe: Gen3 ×4/ ×2/ ×1
- Dedicated 100GE port



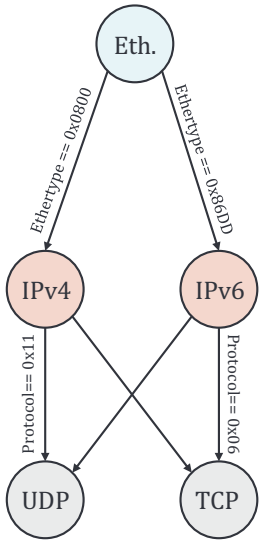
BAREFOOT

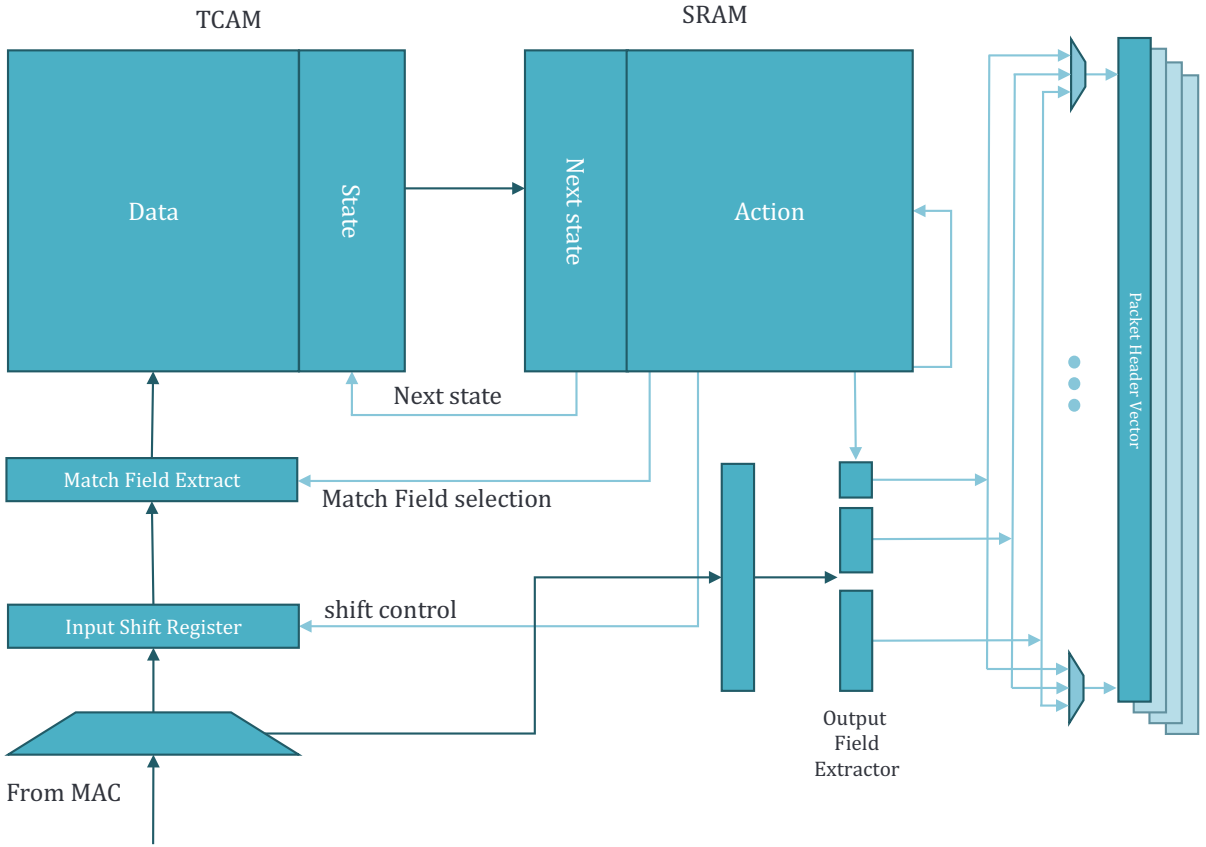
Tofino – Simplified Block Diagram



BAREFOOT

Fully-Programmable Parser

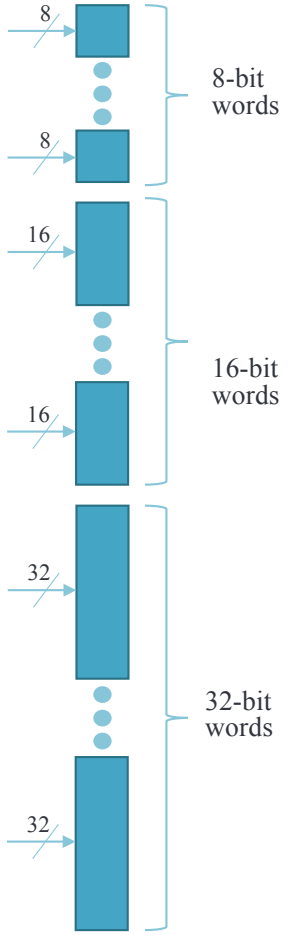




BAREFOOT

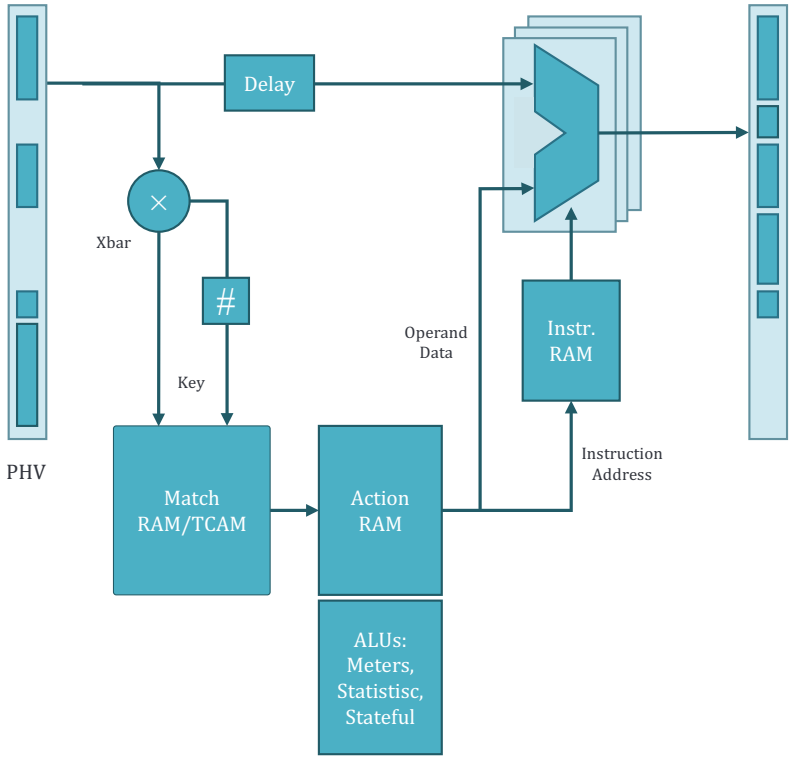
Packet Header Vector (PHV)

- A set of uniform containers that carry the headers and metadata. along the pipeline
- Fields can be packed into any container or their combination.
- Packing is decided by PHV Allocation pass in the compiler.



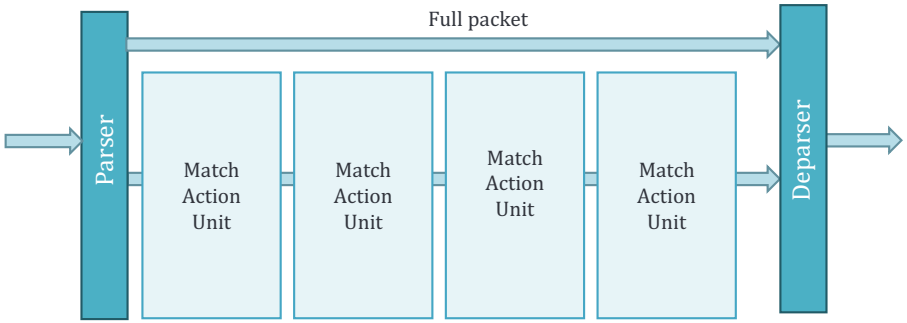
BAREFOOT

Match–Action Unit



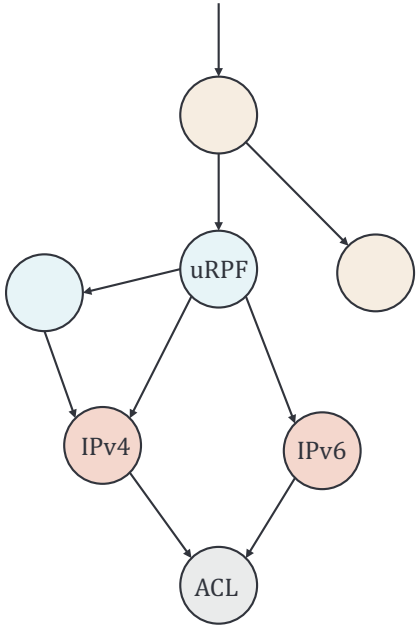
BAREFOOT

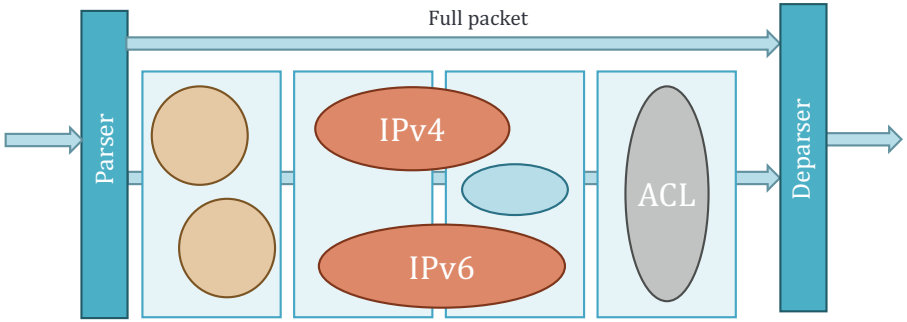
Match–Action Pipeline



BAREFOOT

Match–Action Pipeline

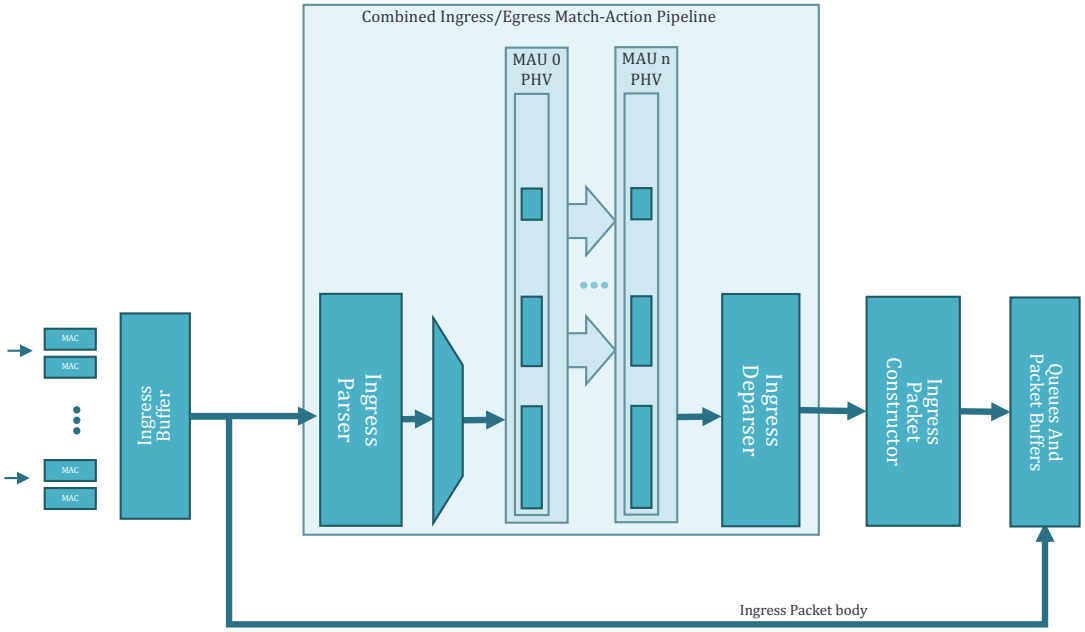




- Multiple small tables per stage
- Large tables spread over multiple stages

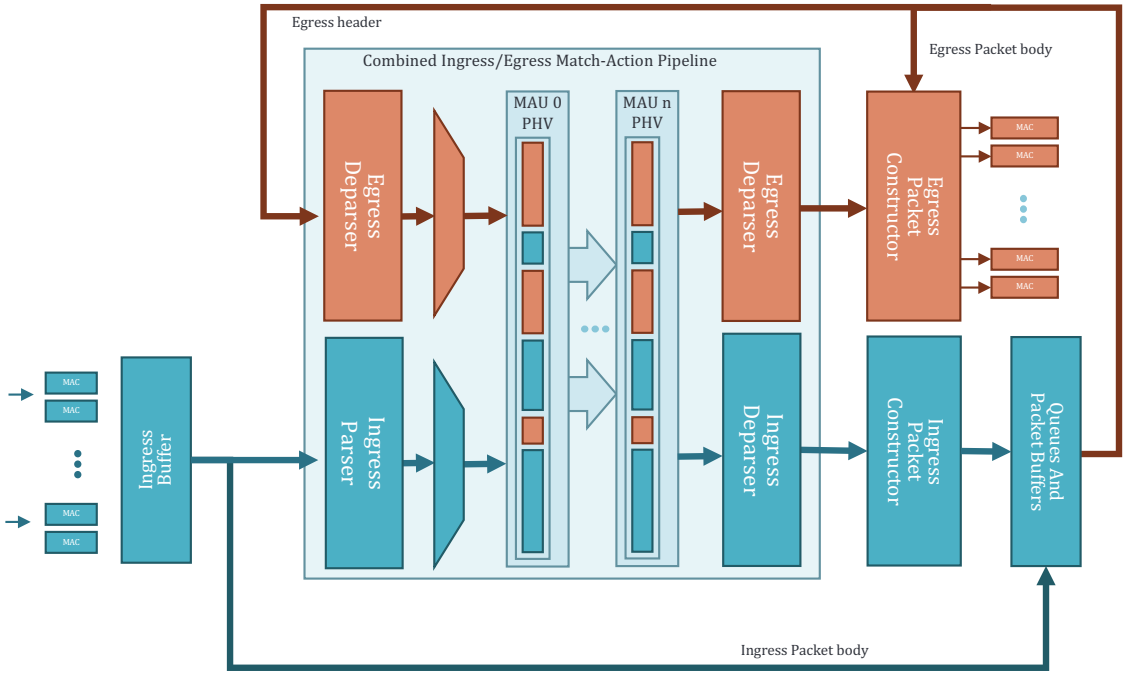
BAREFOOT

Unified Match–Action Pipeline



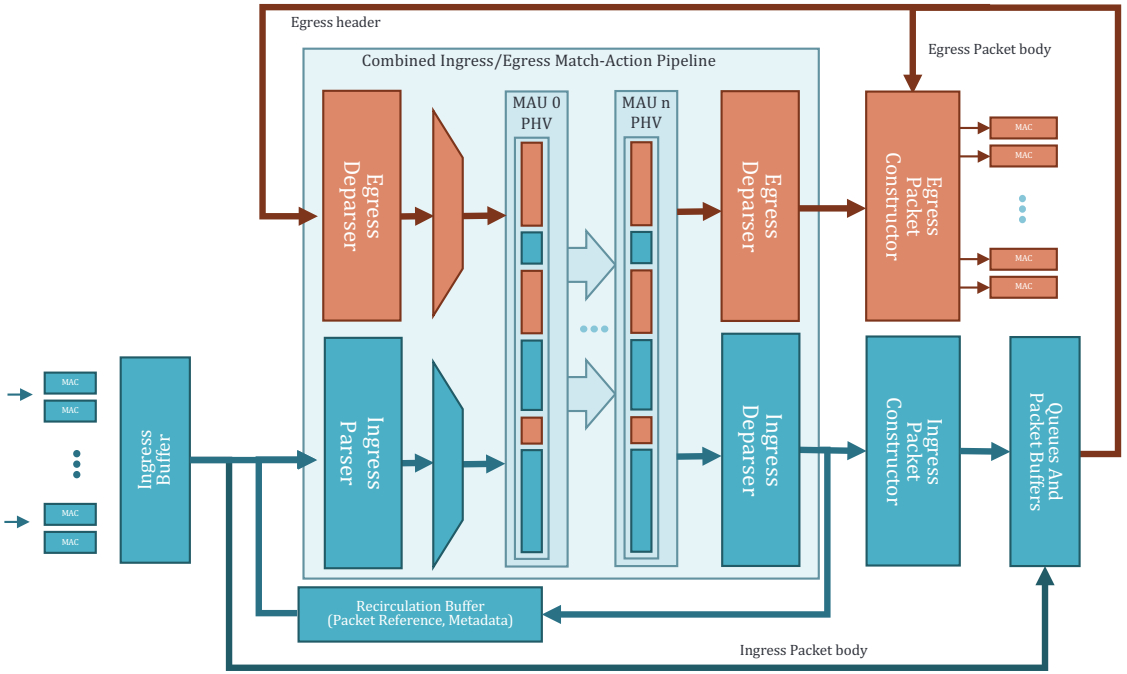
BAREFOOT

Unified Match–Action Pipeline



BAREFOOT

Unified Match–Action Pipeline



BAREFOOT

Programming Tofino

BAREFOOT

Tofino Architecture

- Data plane interfaces
- Intrinsic Metadata

  `ingress_mac_tstamp` – ingress timestamp assigned by MAC

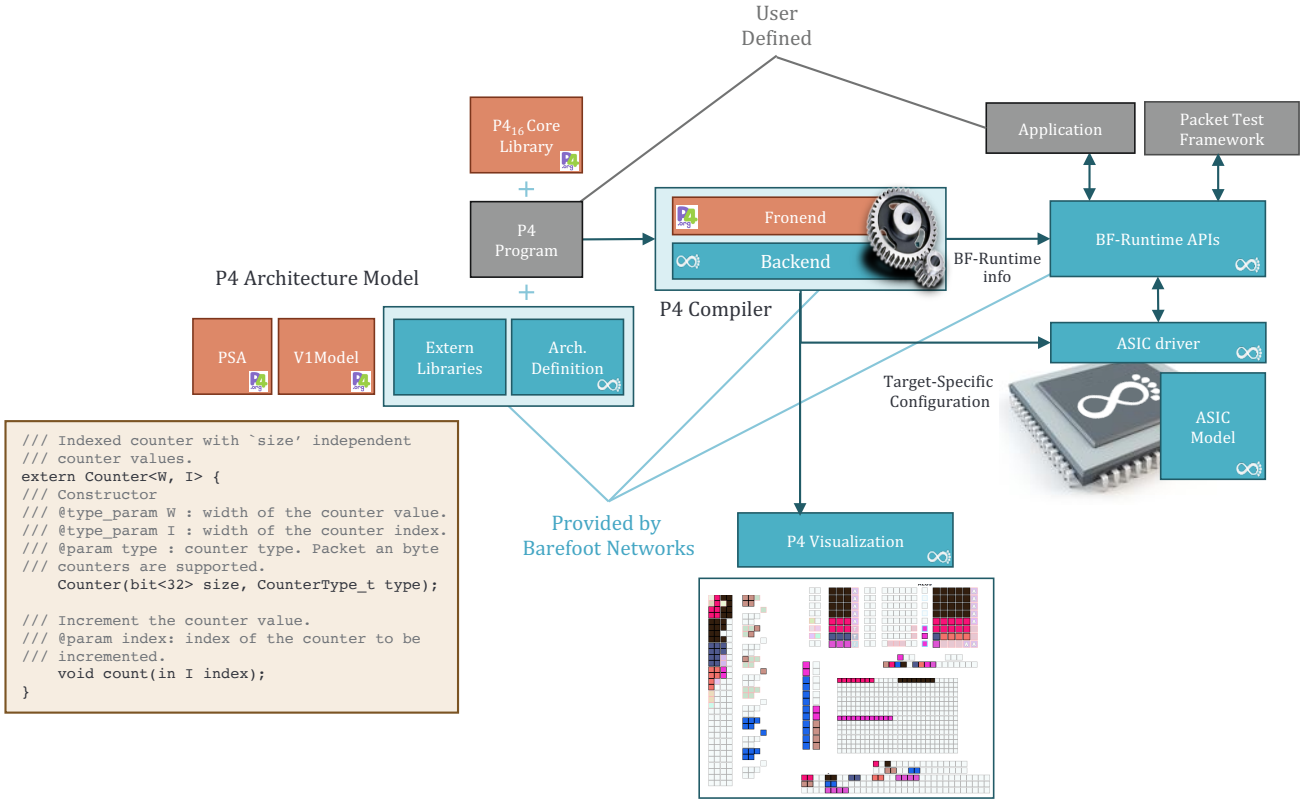
  `ucast_egress_port` – output port
- Tofino-specific Extern

  Registers, Counters, Meters, Filters, etc.

```
/// Register
extern Register<T, I> {
/// Instantiate an array of <size> registers.
/// The initial value is undefined.
    Register(bit<32> size);

/// Initialize an array of <size> registers and set
/// their value to initial_value.
    Register(bit<32> size, T initial_value);

/// Return the value of register at specified index.
    T read(in I index);

/// Write value to register at specified index.
    void write(in I index, in T value);
}
```

BAREFOOT

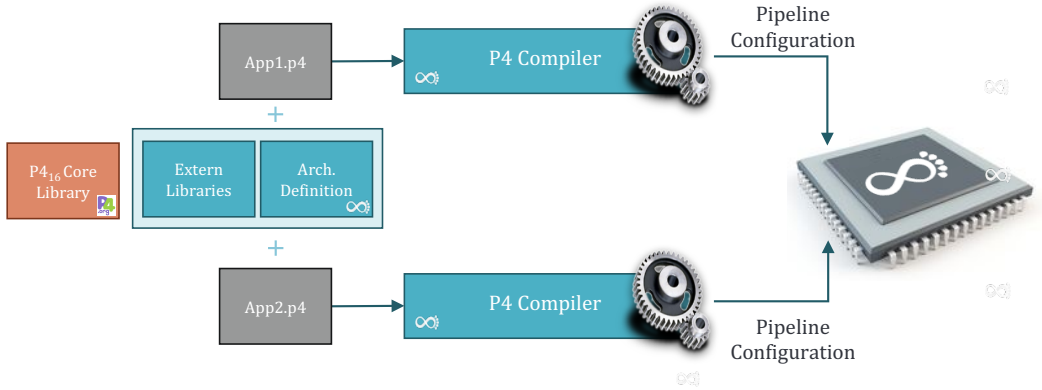
Runtime Configuration




```
/// Indexed counter with `size' independent
/// counter values.
extern Counter<W, I> {
/// Constructor
/// @type_param W : width of the counter value.
/// @type_param I : width of the counter index.
/// @param type : counter type. Packet an byte
/// counters are supported.
    Counter(bit<32> size, CounterType_t type);

/// Increment the counter value.
/// @param index: index of the counter to be
/// incremented.
    void count(in I index);
}
```

BAREFOOT

Multi–Pipeline configuration



BAREFOOT

Thank You

Contact Information

MSHARIF@BAREFOOTNETWORKS.COM

BAREFOOT