CLOSING THE GAP BETWEEN PERFORMANCE AND EFFICIENCY IN PROGRAMMABLE NETWORKING

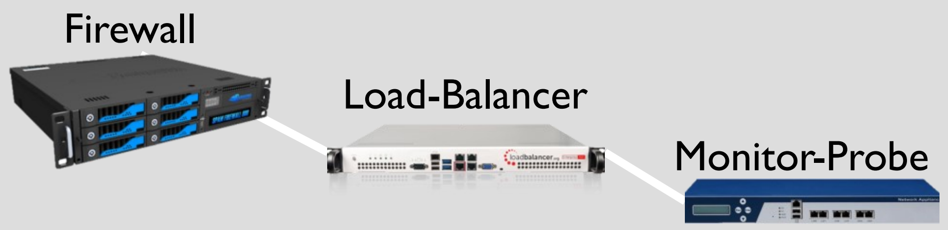
Valerio Bruschi

Doctoral Program in ***Electronics Engineering***

Supervisors	***Prof. Bianchi***
Doctoral Program Coordinator	***Prof. Di Natale***
Academic year	***2021/22***
PhD cycle	***XXXIV***

INTRO: NETWORK PROGRAMMABILITY

- Since 2009, **SDN and NFV** paradigms have emerged:
  - to alleviate *the problem of the ossification of the Internet architecture*
- **SDN**: programmable abstraction, like OpenFlow
  - Too much delegation to the centralized controller
  - Far from being a solution to "all" networking needs
- **NFV**: Network functions redesigned in software and deployed in virtualized networking scenarios
  - Most of the clock cycle spent by the software is in **accessing the memory**;
  - **Large and highly variable** latency.

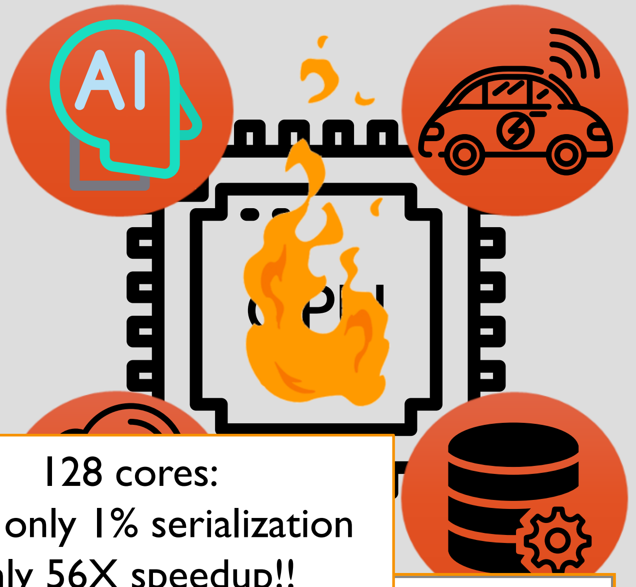


INTRO: CPUS ARE AT A STANDSTILL

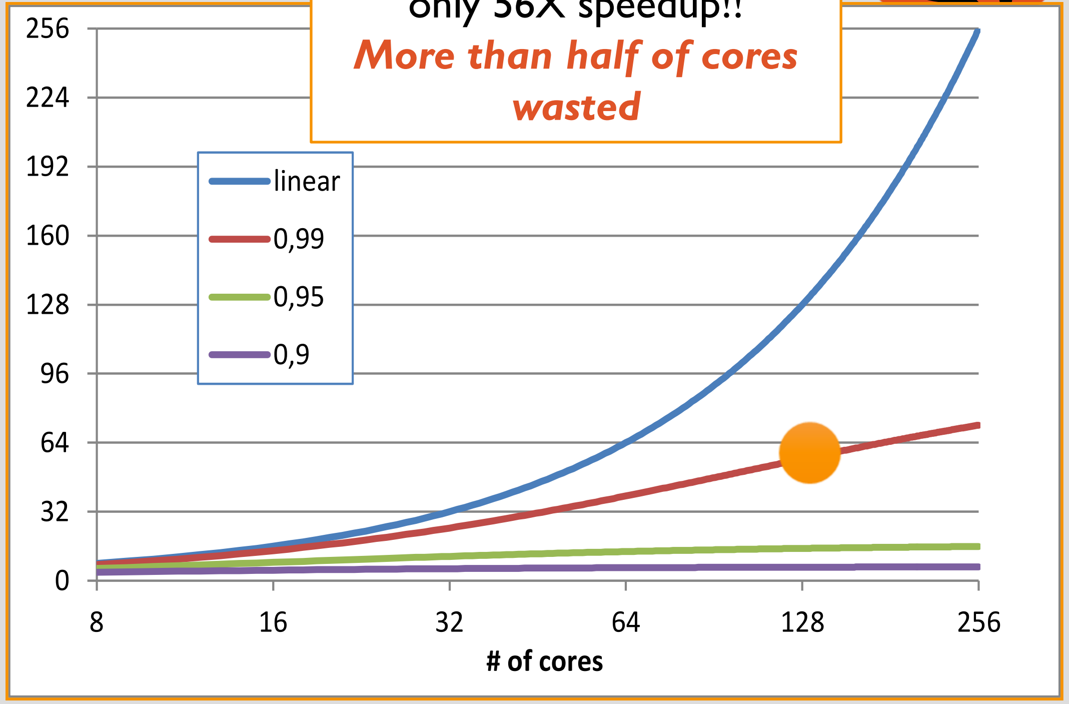
- Moore's law (transistor/chip 2X every 1.5 years) slowing down!
- Dennard Scaling (power/area remains constant) not true since 2007!

- **Power is the bottleneck!**
  - *(downscale chip speed to avoid… **burning it**…!)*

- Amdahl's law Multicore is bounded up by parallelizability
  - $speedup = \frac{1}{(1-F)+\frac{F}{N}}$



128 cores:
With only 1% serialization only 56X speedup!!
More than half of cores wasted



INTRO: BACK TO HW SPECIALIZATION

- HW specialization can provide very efficient and high-performance computation:

  just do few tasks, but extremely well

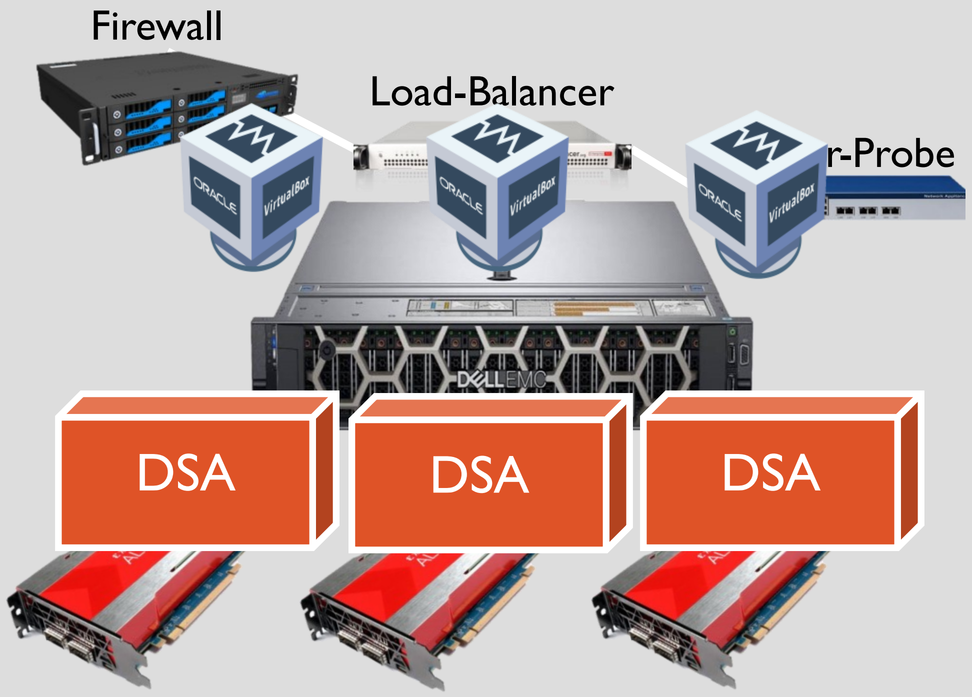
- BUT, the mere availability of these devices is not enough:
  - developing an application for a specific HW-accelerator **involves specific expertise**
  - **high development costs** and **time** that can block the adoption of these technologies.

- ***Network programmability*** may come to the rescue and play an active role:
  - **not only in the typical network-related routines**
  - but also in the computation of some simple (yet meaningful) **upper layer functions**

*Bianchi, G., Faltelli, M., Bruschi, V. "**Back to the Future: Towards Hardware Netputing Architectures (position paper)**," in 2020 MedComNet*

INTRO: NEW HARDWARE-DRIVEN RESEARCH TREND

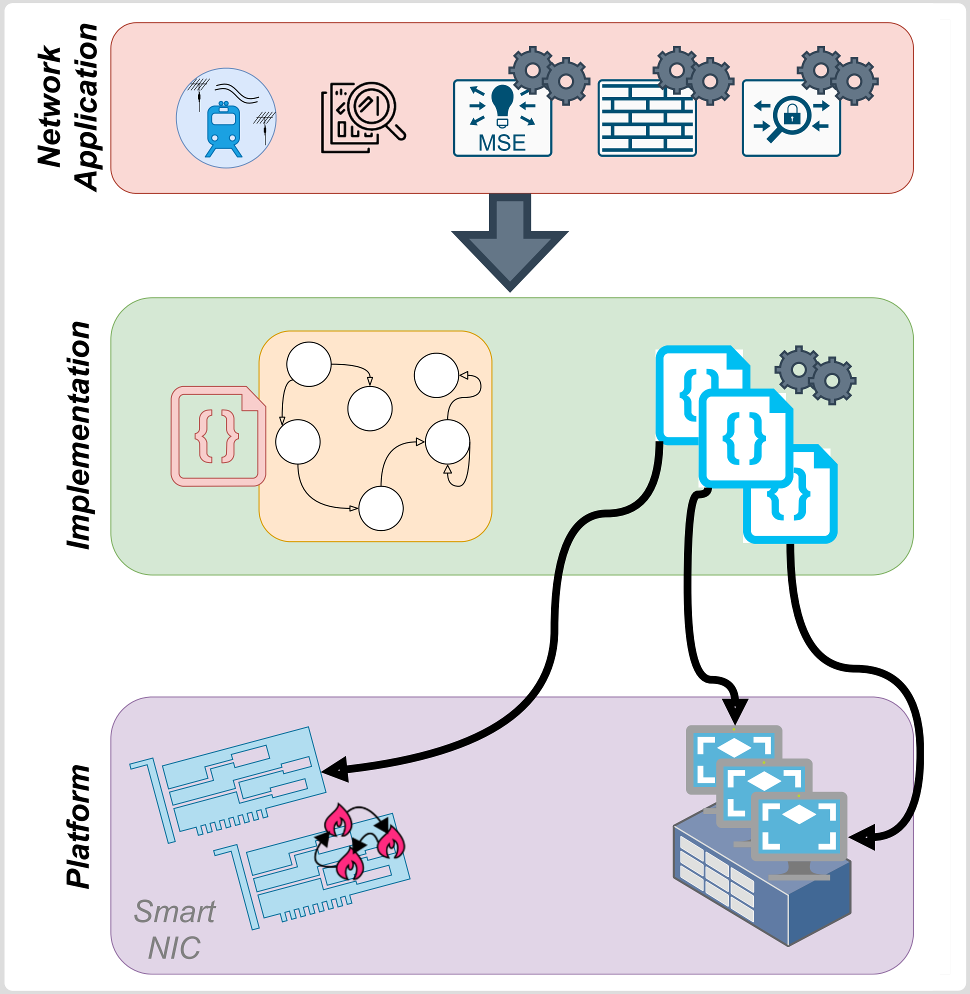
- Born from the lessons learned in SDN and NFV
  - Data plane relies on domain-specific packet processing HW platforms or chipsets
  - to offload network functions to the *Network Interface Card* (NIC)
  - to exploit *HW-compliant programmable abstraction*
    - *e.g. the "PISA" chip (Protocol Independent Switch Architecture) along with the P4 language.*



PHD GOALS

Main goal:
PERFORMANCE + COMPUTING CAPABILITIES + EFFICIENCY

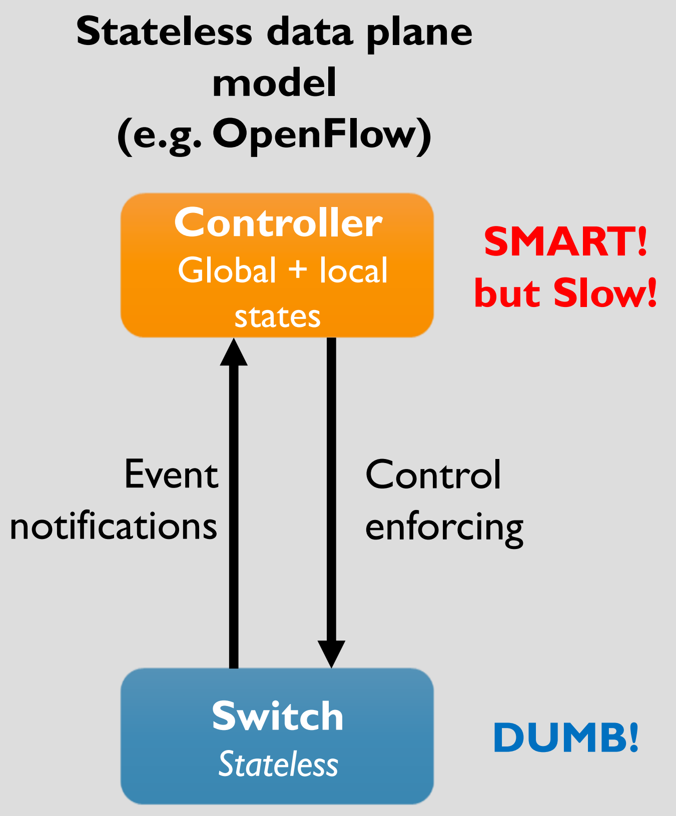
1. With a low-level strategy: **Platforms for Networking**
   - **Propose** a programming abstraction able to program the hardware level for the execution of stateful network functionalities at high speed.
   - **Validate** the proposed abstraction in different use cases and scenarios.
2. With a high-level strategy: **Data structure and Algorithms for Networking**
   - **Design** and **implement** ad hoc solutions tailored to a specific use case to improve performance and efficiency.
   - **Validate** the approach through two fundamental Network Functions: packet classification and network monitoring by per flow distinct counting.

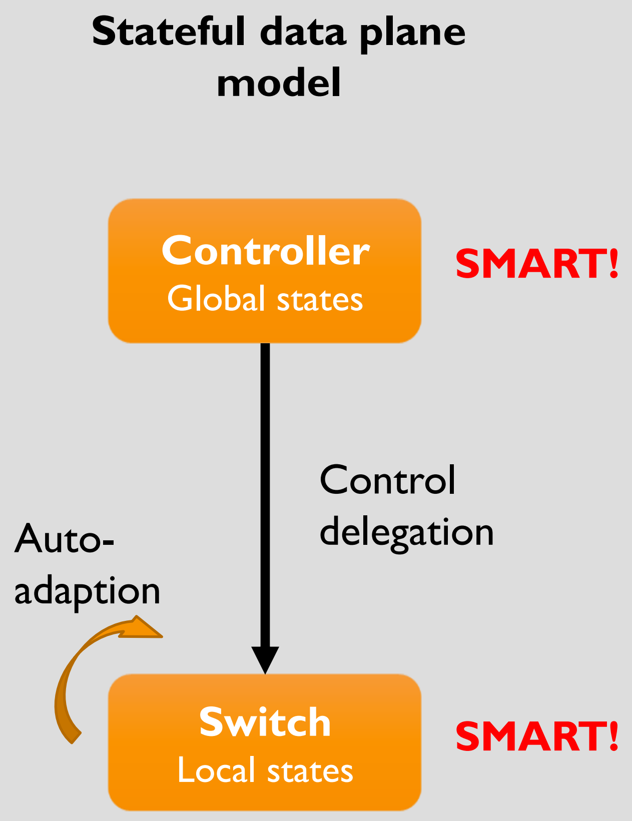


PLATFORMS FOR NETWORKING

STATEFUL FLOW PROCESSING CHALLENGE

- Typical **stateless** data plane:
  - requires the **intervention** of the **controller** for any change of the forwarding decision
- A **stateful** strategy refers to:
  - keep and manipulate persistent states **locally**
  - significantly **reduce** the interaction between switches and the controller
  - **self-adapt** the forwarding behavior according to network events





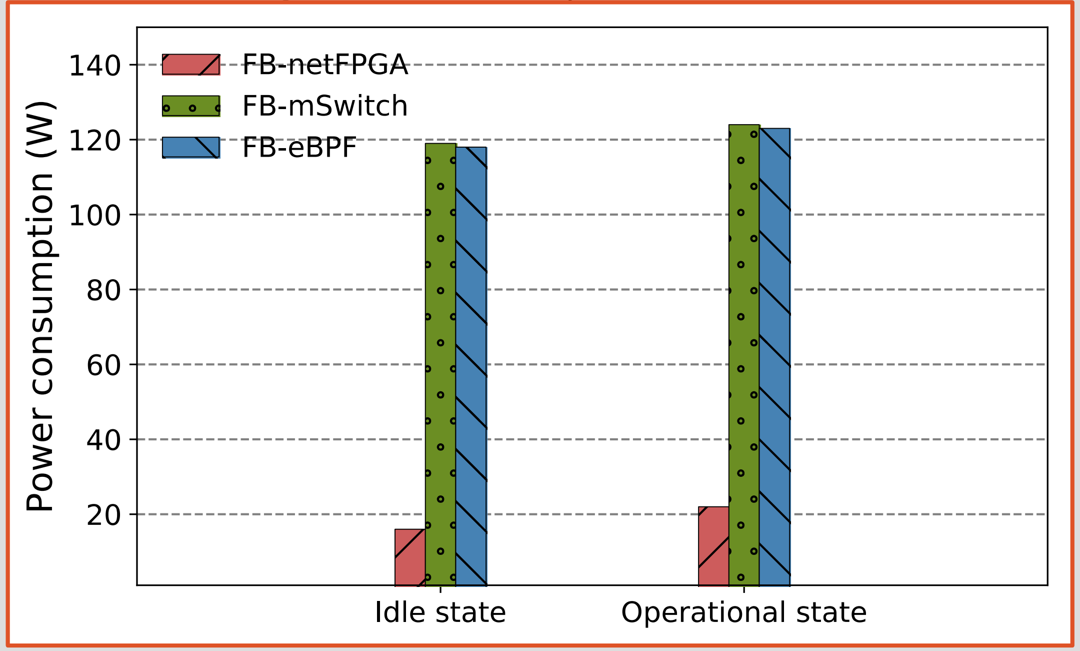
A NEW STATEFUL PROGRAMMABLE ABSTRACTION

We propose FlowBlaze, an eXtended Finite State Machine (XFSM) executor

- **XFSMs** appear to be very expressive as low-level abstraction
  - Natural model to describe a stateful process
  - Ability to specify and compute a wide (and programmable) class of stateful information
  - Efficient storage and management of per-flow stateful information
  - Suitable for hardware offloading

- «Code-once-port-everywhere»
  - Platform independent abstraction
  - Seamless portability between SW and HW platforms

Application

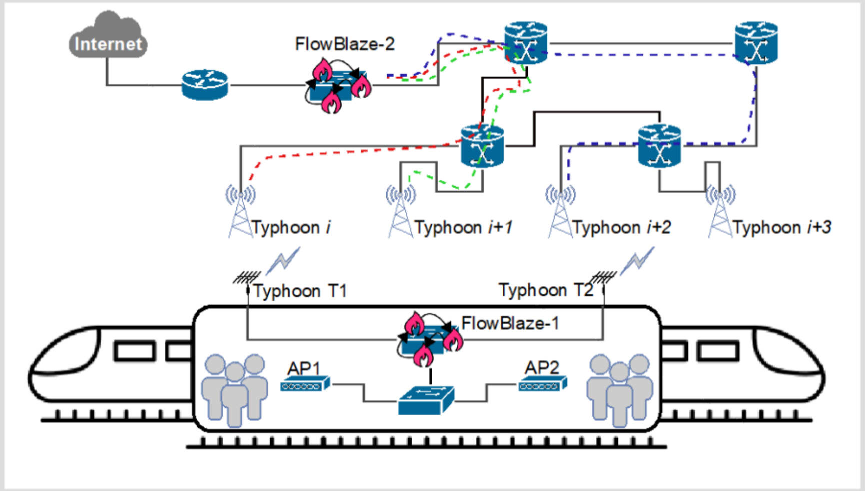


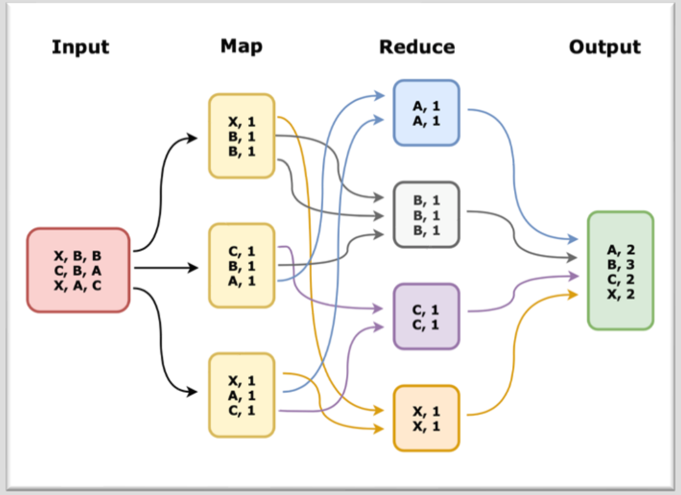
5x gain

TARGETS "COMPLEX" NETWORK FUNCTIONS AND BEYOND

MOBILITY MANAGEMENT TO **ENSURE SESSION CONTINUITY** DIRECTLY IN THE DATA PLANE

DATA AGGREGATION IN THE DATA PLANE THROUGH **ONLINE MAPREDUCE TASKS OFFLOADING**





Event handled in datapath!! update forwarding rules in **1 packet time**

***3 ns** @ 40B x 100 Gbps || **5 ns** @ 64B x 100 Gbps*

1. *Implemented in a real railway operated by **Ferrocarrils de la Generalitat de Catalunya** as demo of the H2020 5G-PICTURE project.*
2. *Bruschi, V., et al., "**Ensuring Session Continuity for Railways using a Stateful Programmable Dataplane**," in 2021 IEEE 5G for CAM.*
3. *Zou, J., Legg, P., Santiago, R., Bruschi, V., et al., "**Europe's First 5G-Ready Railway Trial Utilizing Integrated Optical Passive WDM Access and Broadband Millimeter-Wave to Deliver Multi-Gbit/s Seamless Connectivity**," in 2020 IEEE ECOC*
4. *Bruschi, V., et al., "**Offloading Online MapReduce tasks with Stateful Programmable Data Planes**," in 2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN) (pp. 17-22). IEEE.*

DATA STRUCTURES & ALGORITHMS FOR NETWORKING

CARDINALITY ESTIMATION: THE PROBLEM

Objective:

Find the number of distinct elements in a data stream with repeated elements.

Use case: *find* ***scan-type flows****, namely flows which exhibit a large cardinality in terms of number of distinct source/destination addresses, or in most generality packet-level identifiers (e.g. ports, header fields, etc).*

Challenges:

- It must somehow **"remember"** the observed elements for duplicate removal
- while measuring a flow size only needs a counter

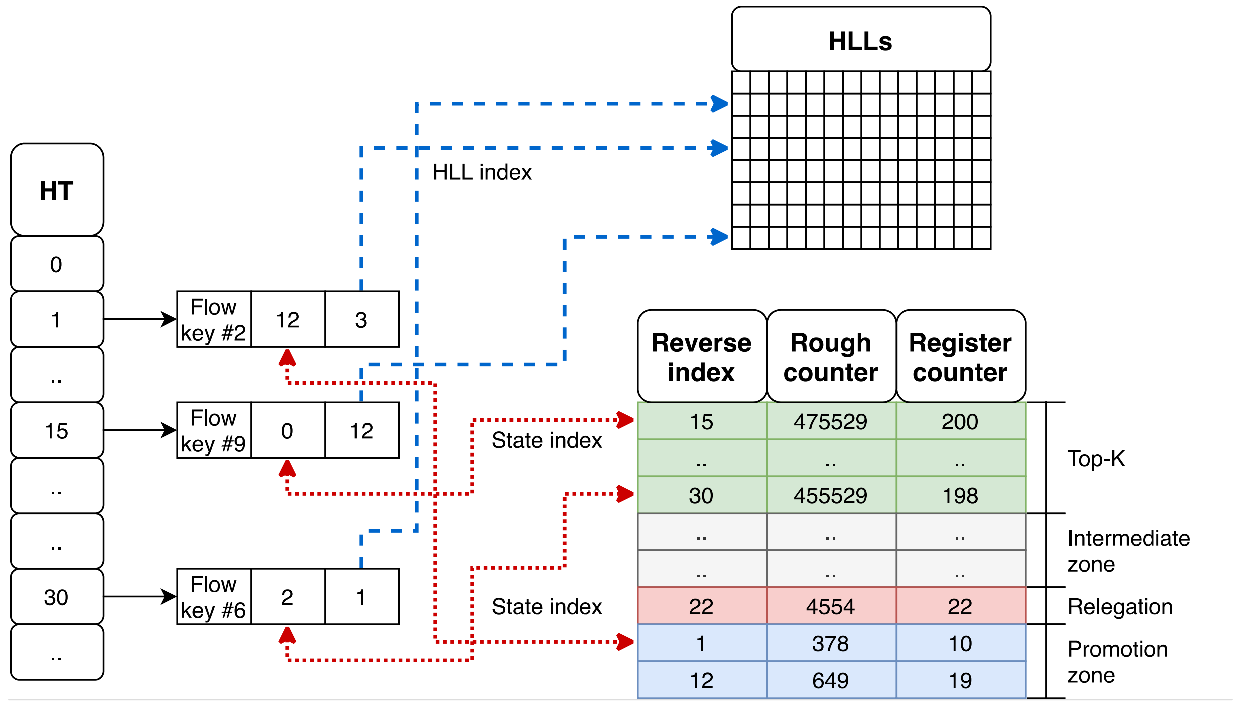
Even more challenging if used to estimate top-k flows:

- Using an HyperLogLog for each source to monitor requires a huge amount of memory
- Standard methods used for top-k selection **do not work** in the case of cardinality estimation

FLOWFIGHT: TOP-k CARDINALITY ESTIMATION

- **Cardinality estimator**: HyperLogLog sketch
  - We restrict the available HLLs to a number slightly higher than *k*

- **Top-K Data structure** inspired by Stream Summary
  - **cheaply** updates the HLL sketch for monitored
  - **easily** identifies the flow with the lowest cardinality to kick out
    - A rough estimation of the cardinality is performed for each flow



Reverse index	Rough counter	Register counter
15	475529	200
..	..	..
30	455529	198
..	..	..
..	..	..
22	4554	22
1	378	10
12	649	19

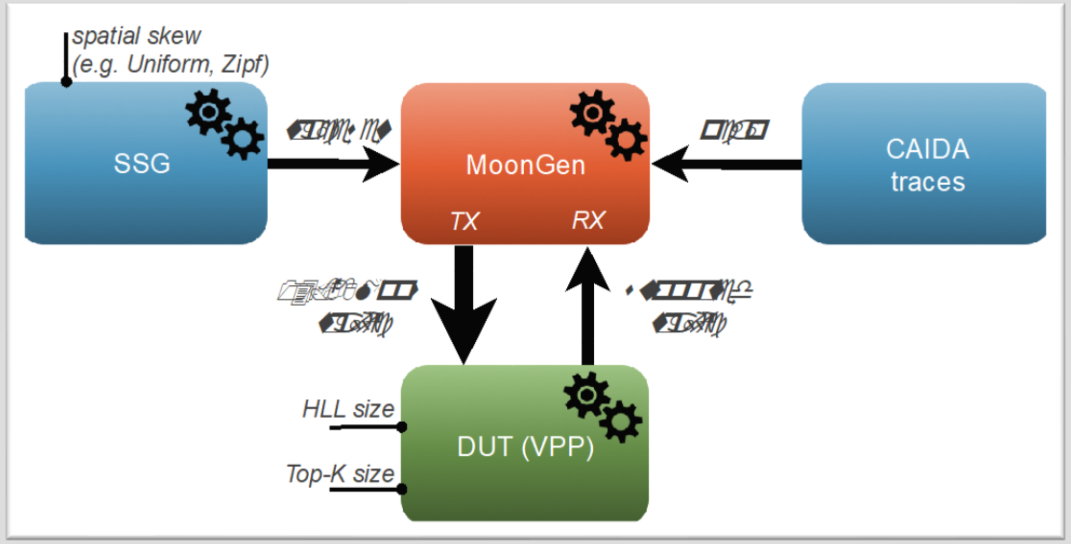
- **Randomized Access Policy** (RAP)
  - we propose an innovative randomized access policy based on a "fight" between flows.

*Bruschi, V., Pontarelli, S., Tollet, J., Barach, D., Bianchi, G. "**FlowFight: High performance-low memory top-k spreader detection**," in Elsevier Computer Networks*

EXPERIMENTAL EVALUATIONS IN ACTUAL NETWORK DEPLOYMENTS

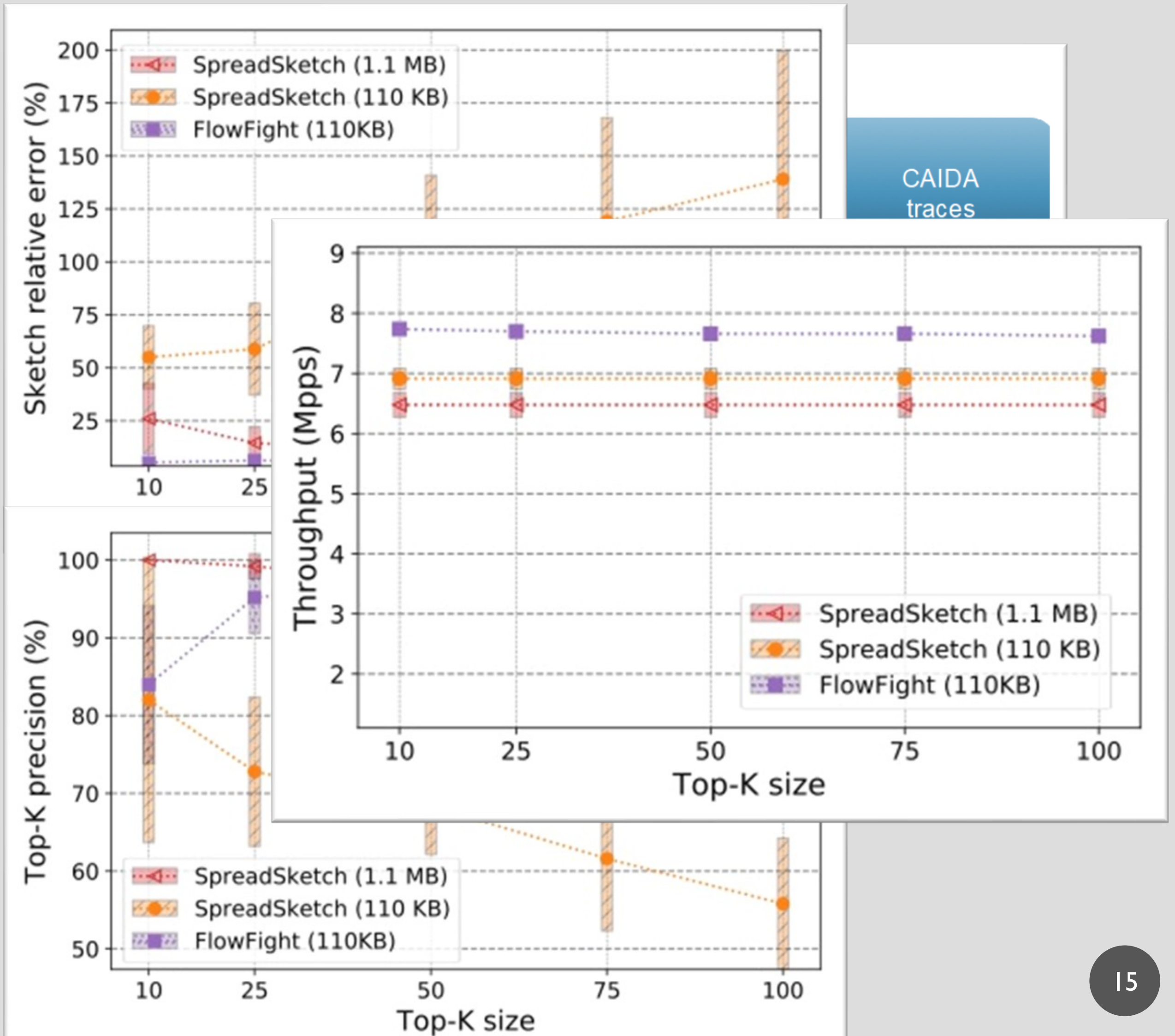
- Algorithm validation in actual network deployment:
  - Implemented in a software router VPP

- SpreadSketch, state-of-the-art for top-k spreaders detection
  - two configurations according to its memory occupancy
    - 4 x 512 = 110 KB (as FlowFight)
    - 4 x 4096 = 1.1 MB



FLOWFIGHT VS SPREADSKETCH

- Algorithm validation in actual network deployment:
  - Implemented in a software router VPP
- SpreadSketch, state-of-the-art for top-k spreaders detection
  - two configurations according to its memory occupancy
    - 4 x 512 = 110 KB (as FlowFight)
    - 4 x 4096 = 1.1 MB
- FlowFight uses **10x times less** memory
- FlowFight achieves **higher** throughput



CONCLUSIONS

- Two strategies to close the gap:
  - **HW specialization** can provide very efficient and high-performance computation
    - Defining (domain-specific) programming abstractions could bring a critical boost in the DSAs deployment
  - **Ad-hoc solutions** (sketches, data structures and algorithms) to improve performance and resource efficiency

- We hope that the combination of the proposed strategies can lay the foundation for a new model
  - For both **packet processing** as well as **application-level acceleration**
  - Exploiting the significant throughput and latency improvements provided by *Network Programmability innovations*
  - While *reducing the power dissipation* of future cloud deployments

THANKS!

Q&A