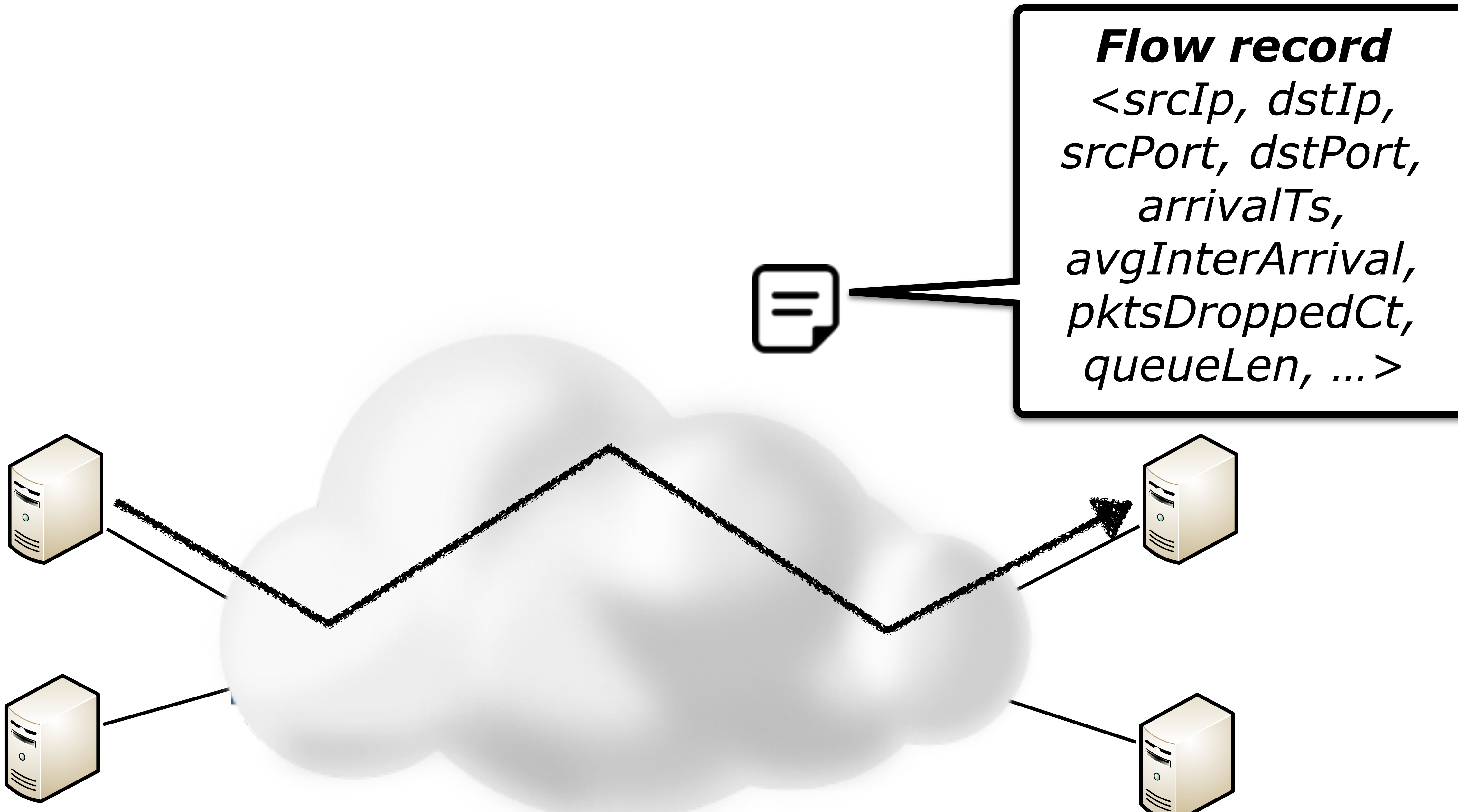
TurboFlow: Information Rich Flow Record Generation on Commodity Switches

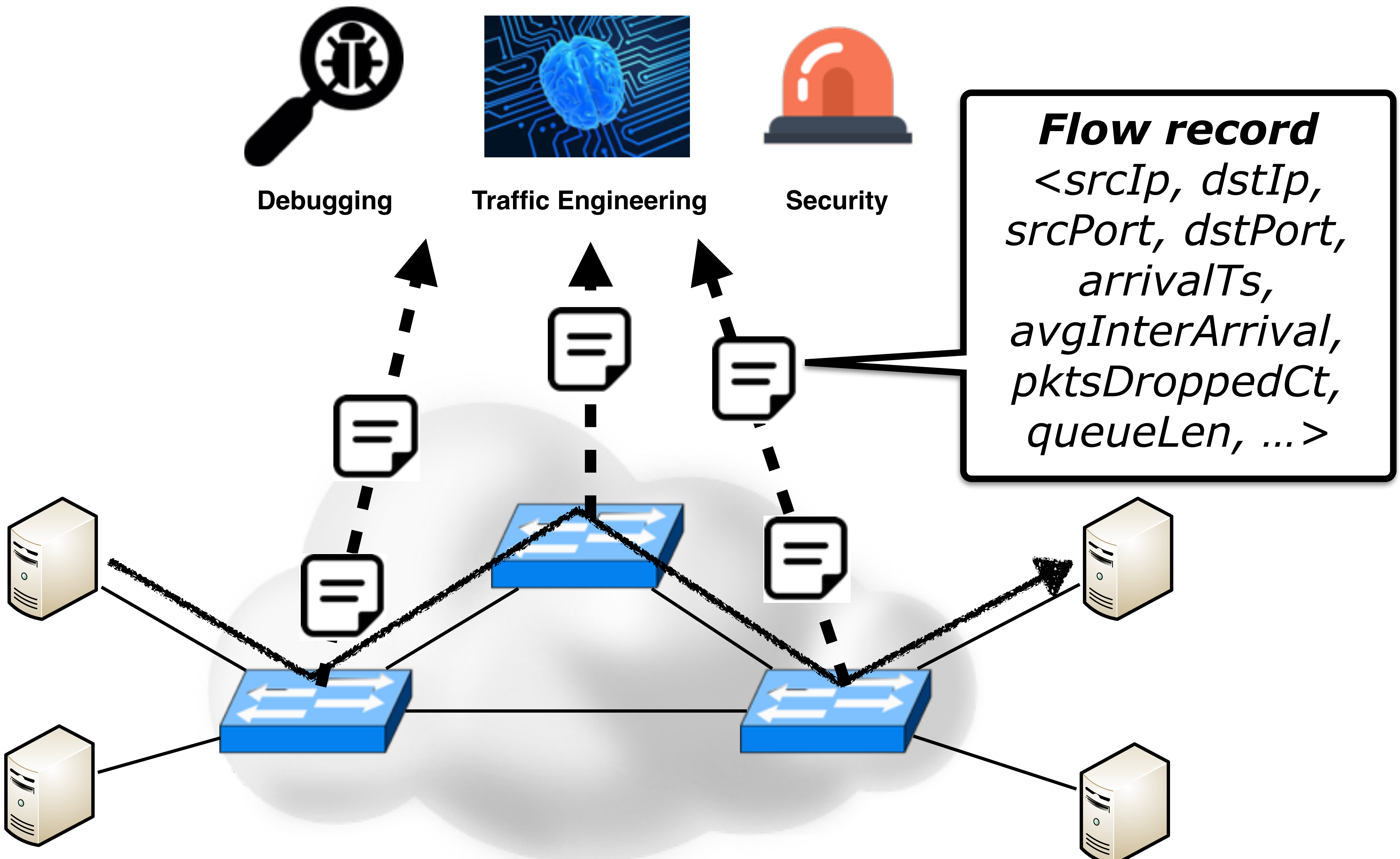
John Sonchack[1], Adam J. Aviv[2], Eric Keller[3], Jonathan M. Smith[1]

[1]University of Pennsylvania, [2]USNA, [3]University of Colorado

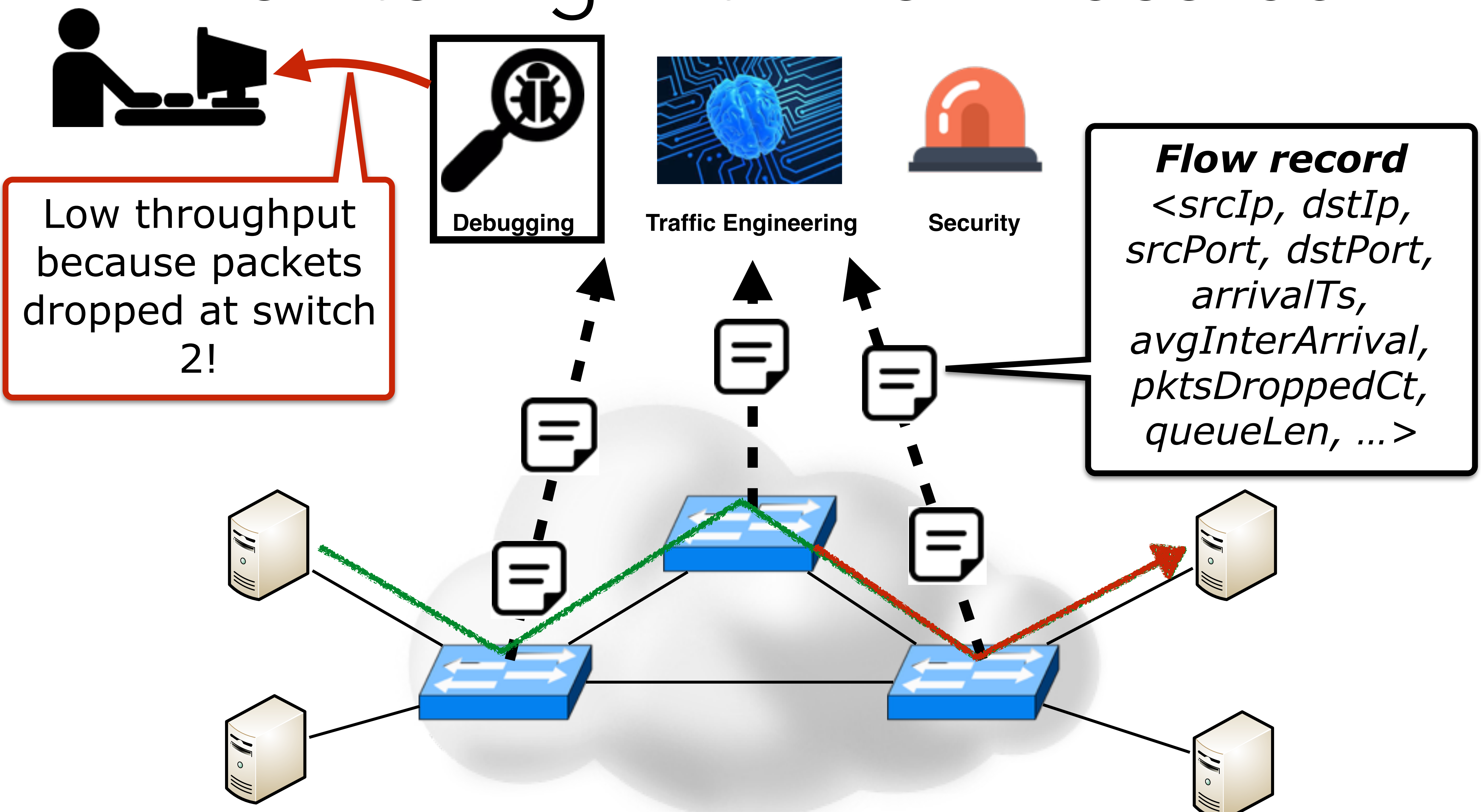
Introduction: Network Monitoring with Flow Records



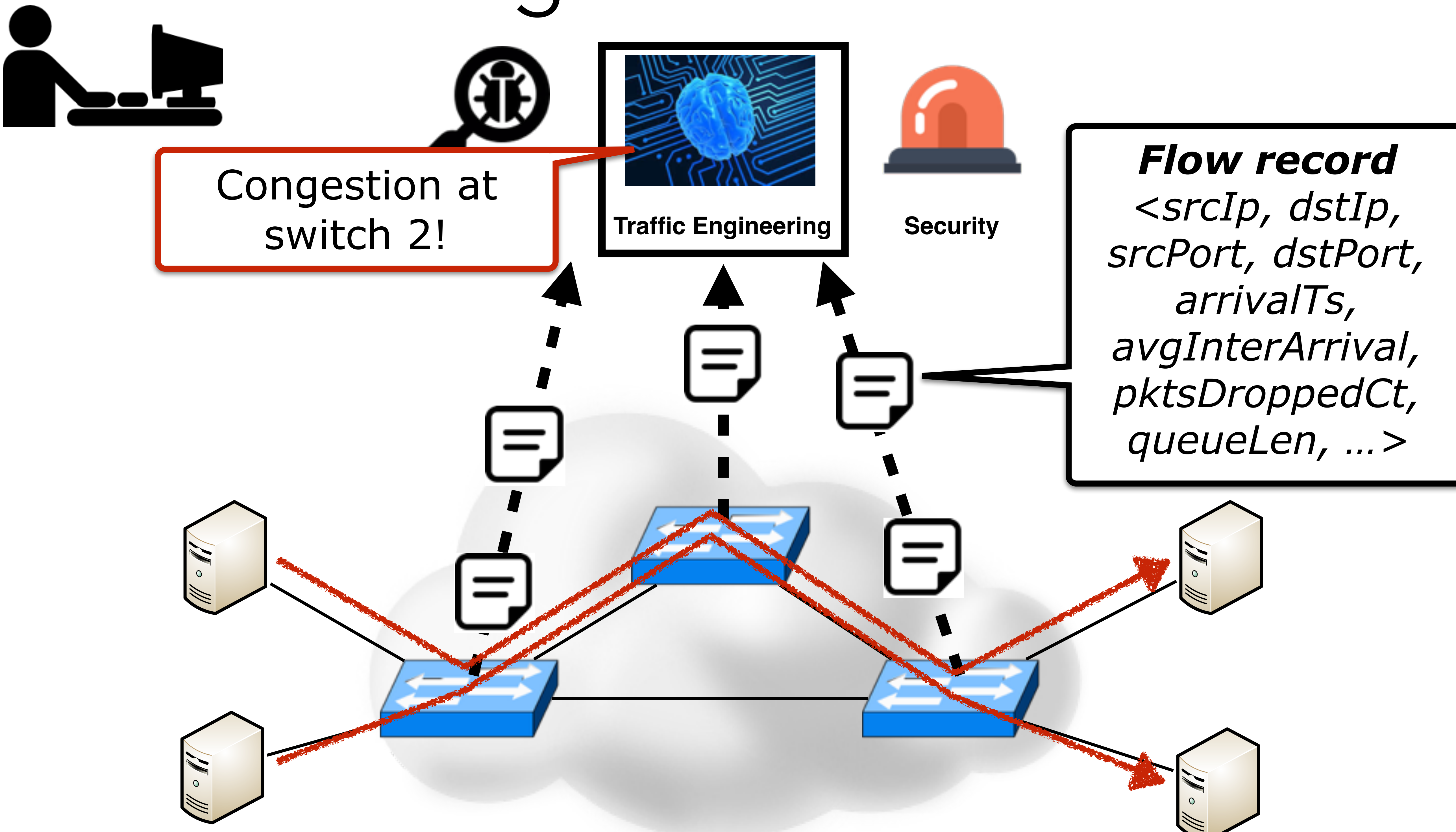
Introduction: Network Monitoring with Flow Records



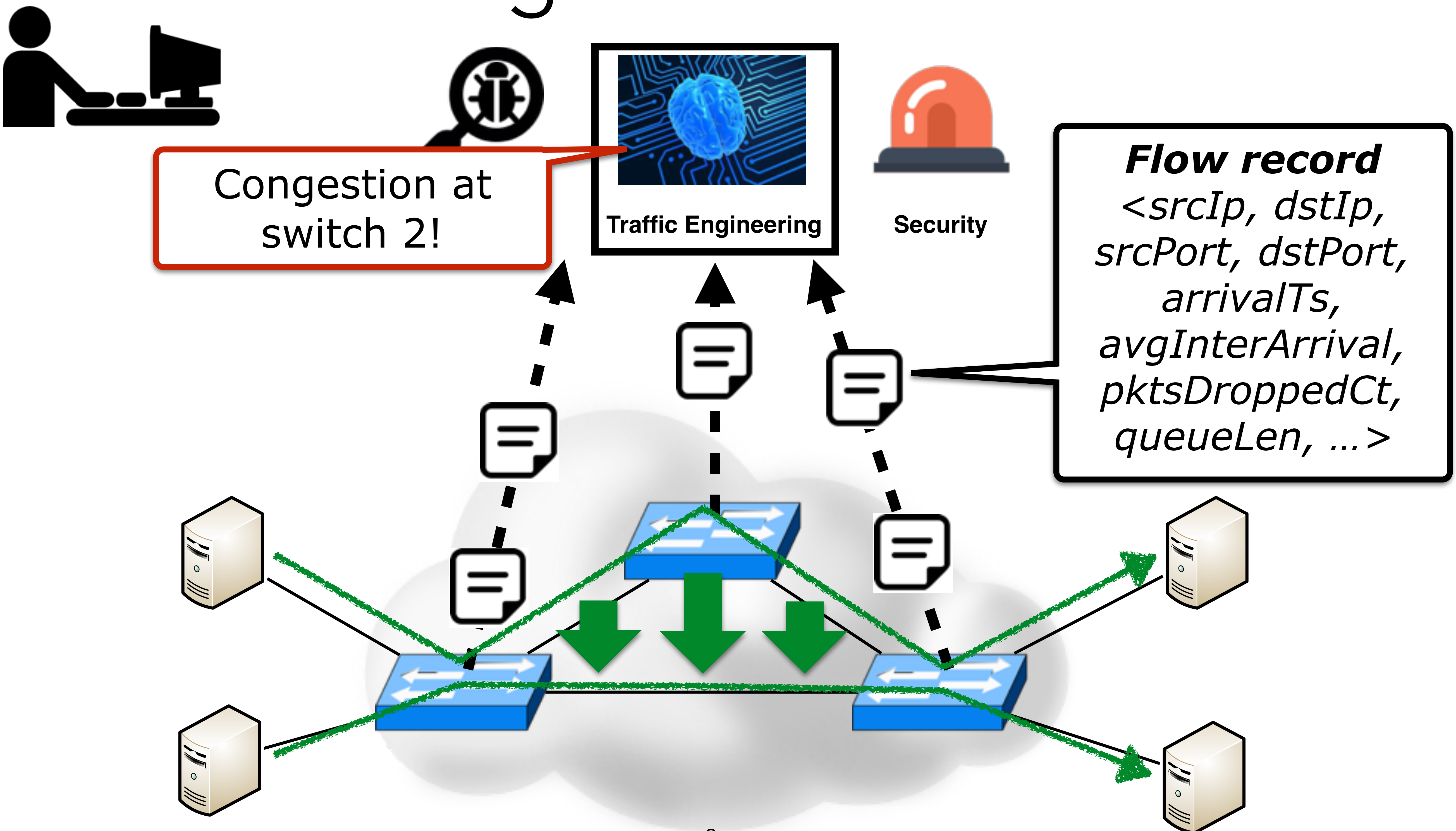
Introduction: Network Monitoring with Flow Records



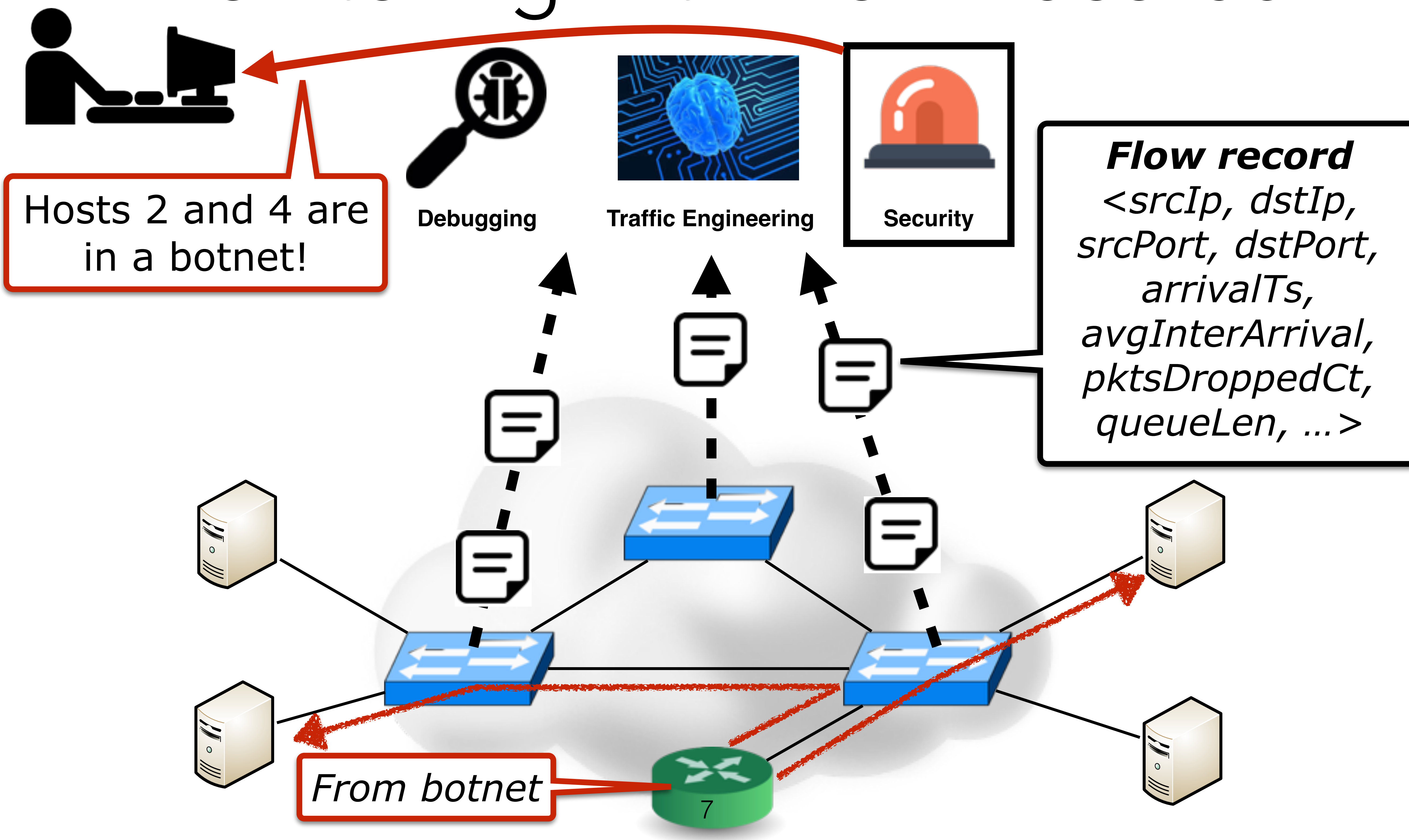
Introduction: Network Monitoring with Flow Records



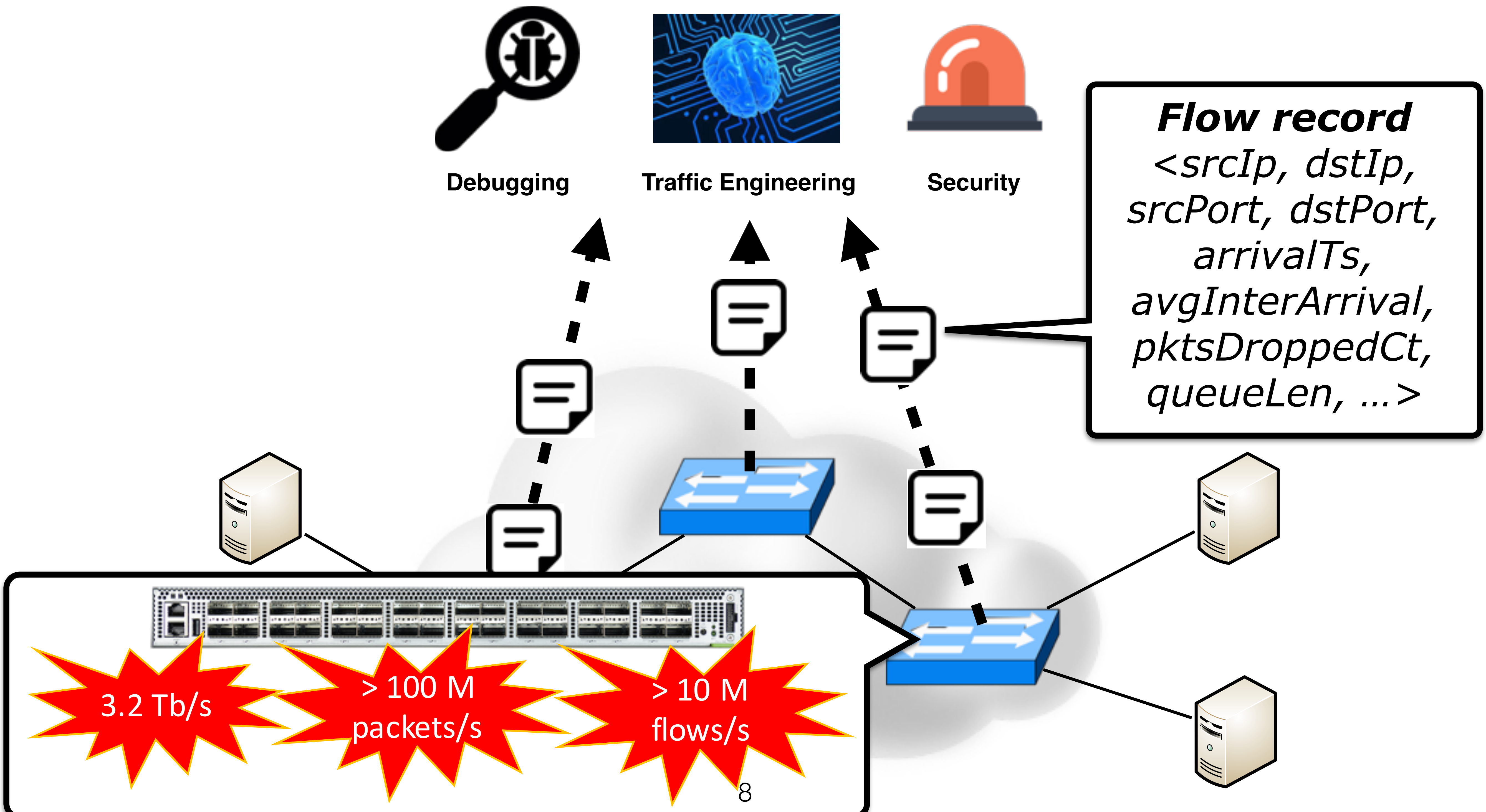
Introduction: Network Monitoring with Flow Records



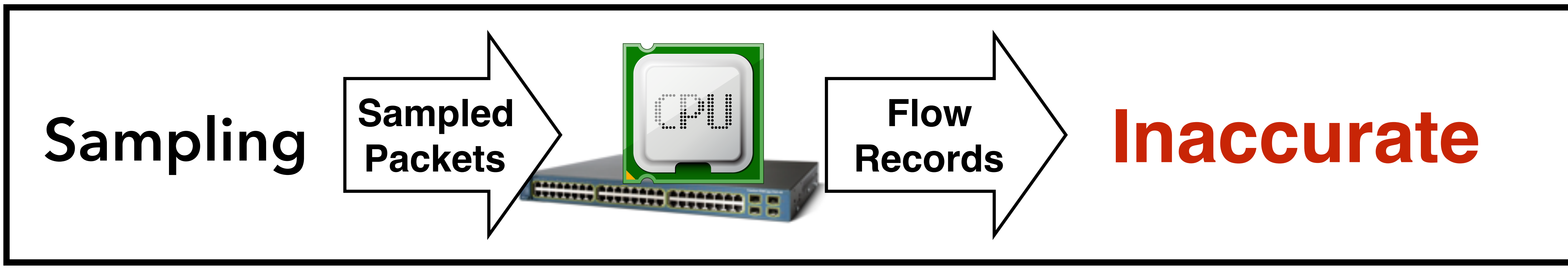
Introduction: Network Monitoring with Flow Records



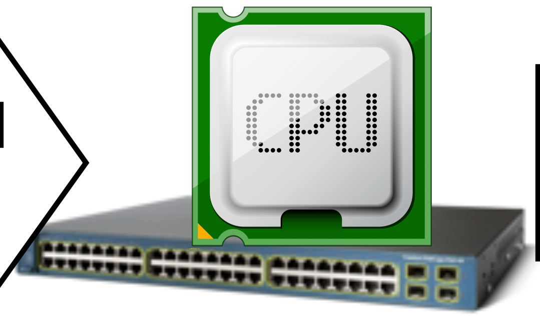
Introduction: Network Monitoring with Flow Records



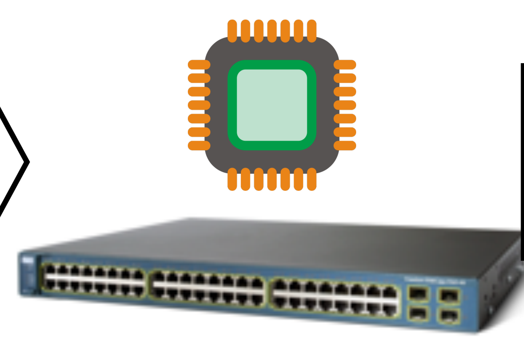
Flow Monitoring Switches: Prior Work



Flow Monitoring Switches: Prior Work

Sampling → Sampled Packets →  → Flow Records → **Inaccurate**

Server Offloading → Packets (or other records) →  → Flow Records → **Expensive**

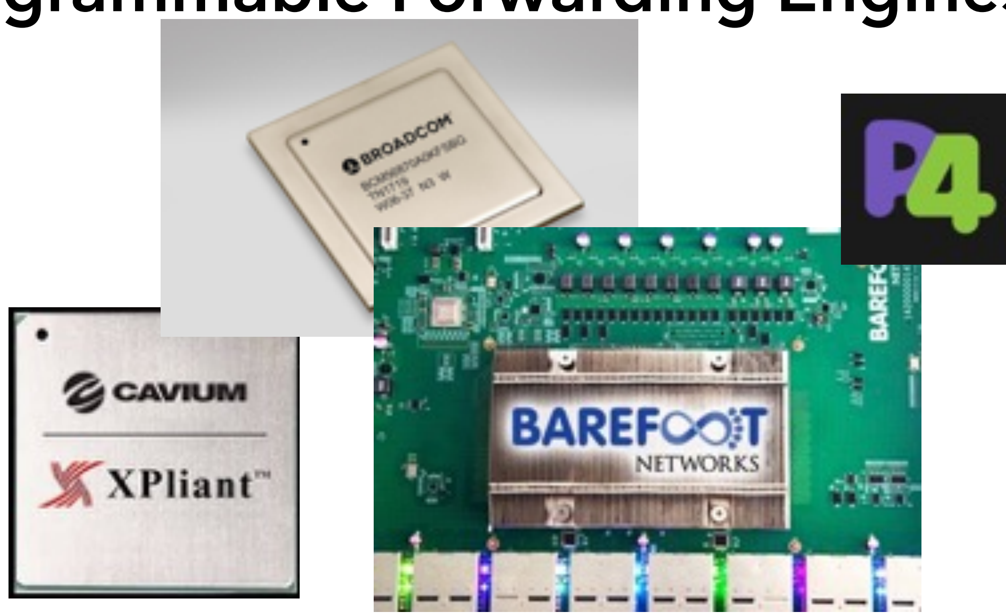
Custom Hardware Offloading → Packets (or other records) →  → Flow Records → **Restrictive**

Introduction: TurboFlow

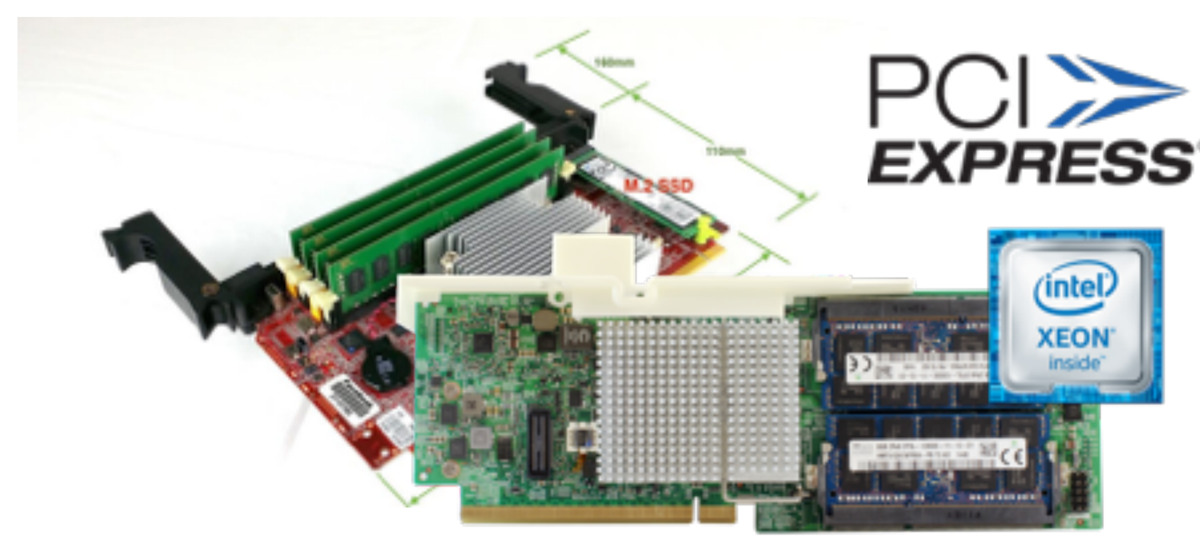
Main idea: *Optimize instead of offload.*

Q : What can we get out of the programmable hardware in next-generation commodity switches?

Programmable Forwarding Engines



Onboard Microservers



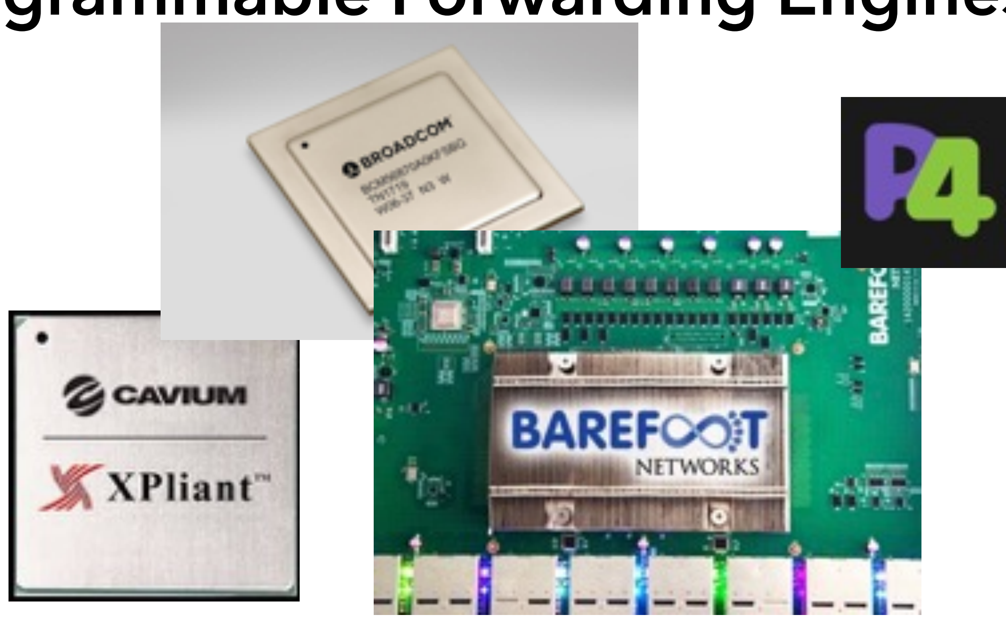
Introduction: TurboFlow

Main idea: *Optimize instead of offload.*

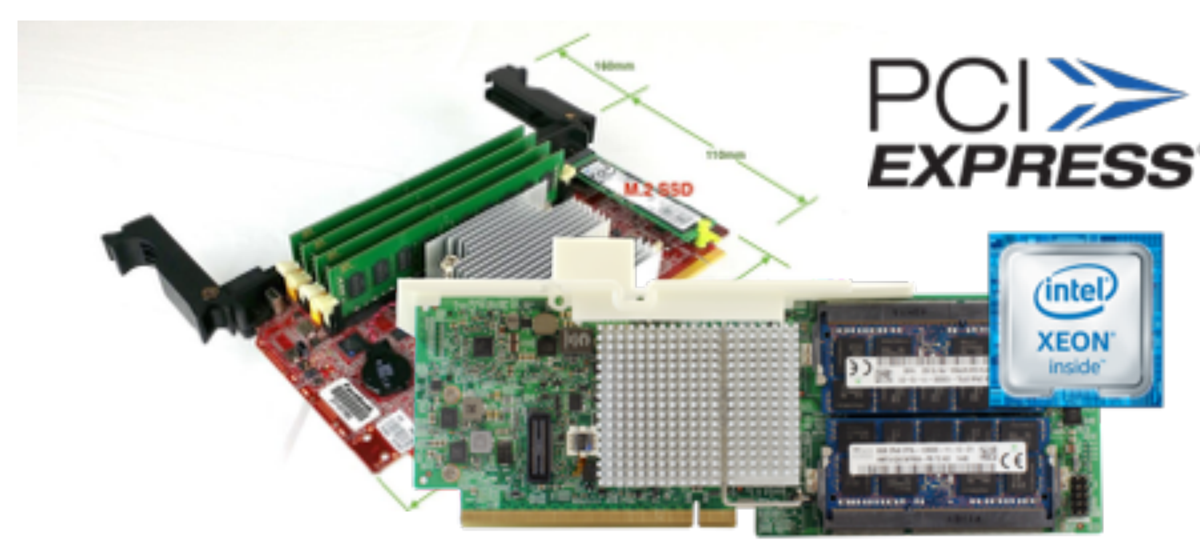
Q : What can we get out of the programmable hardware in next-generation commodity switches?

A : Flow record generation for **multi-terabit** rate traffic **without sampling or offloading**.

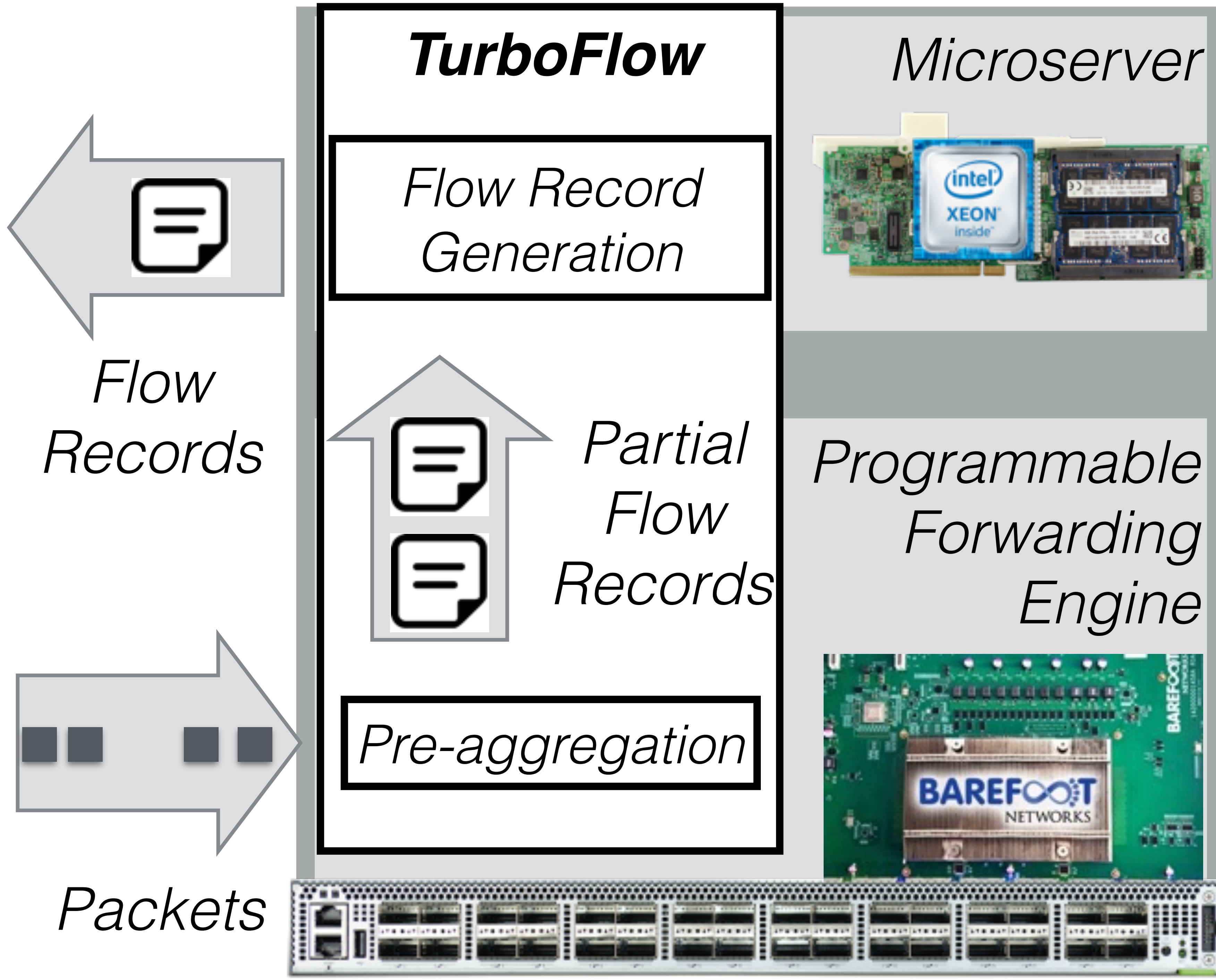
Programmable Forwarding Engines



Onboard Microservers

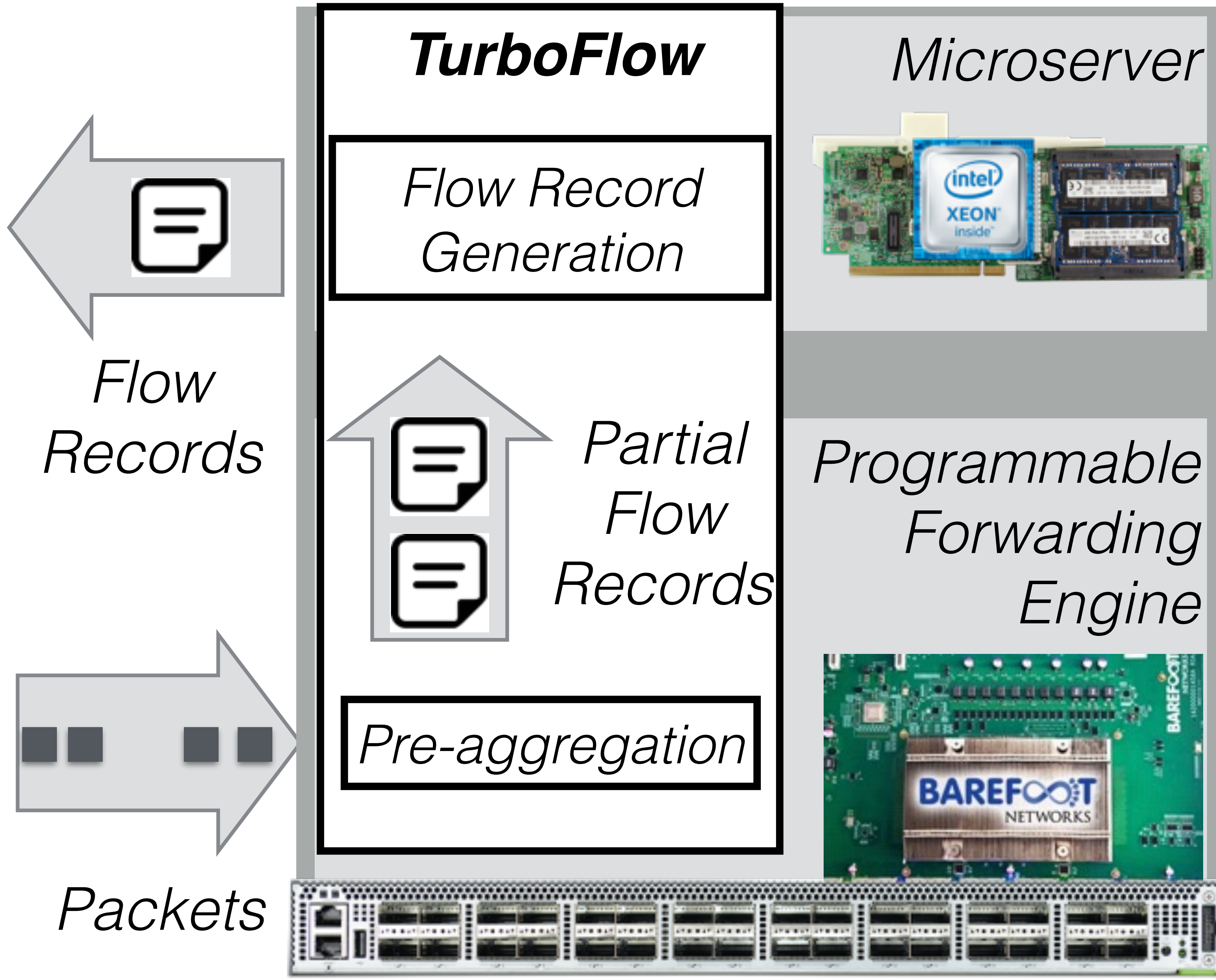


Introduction: TurboFlow

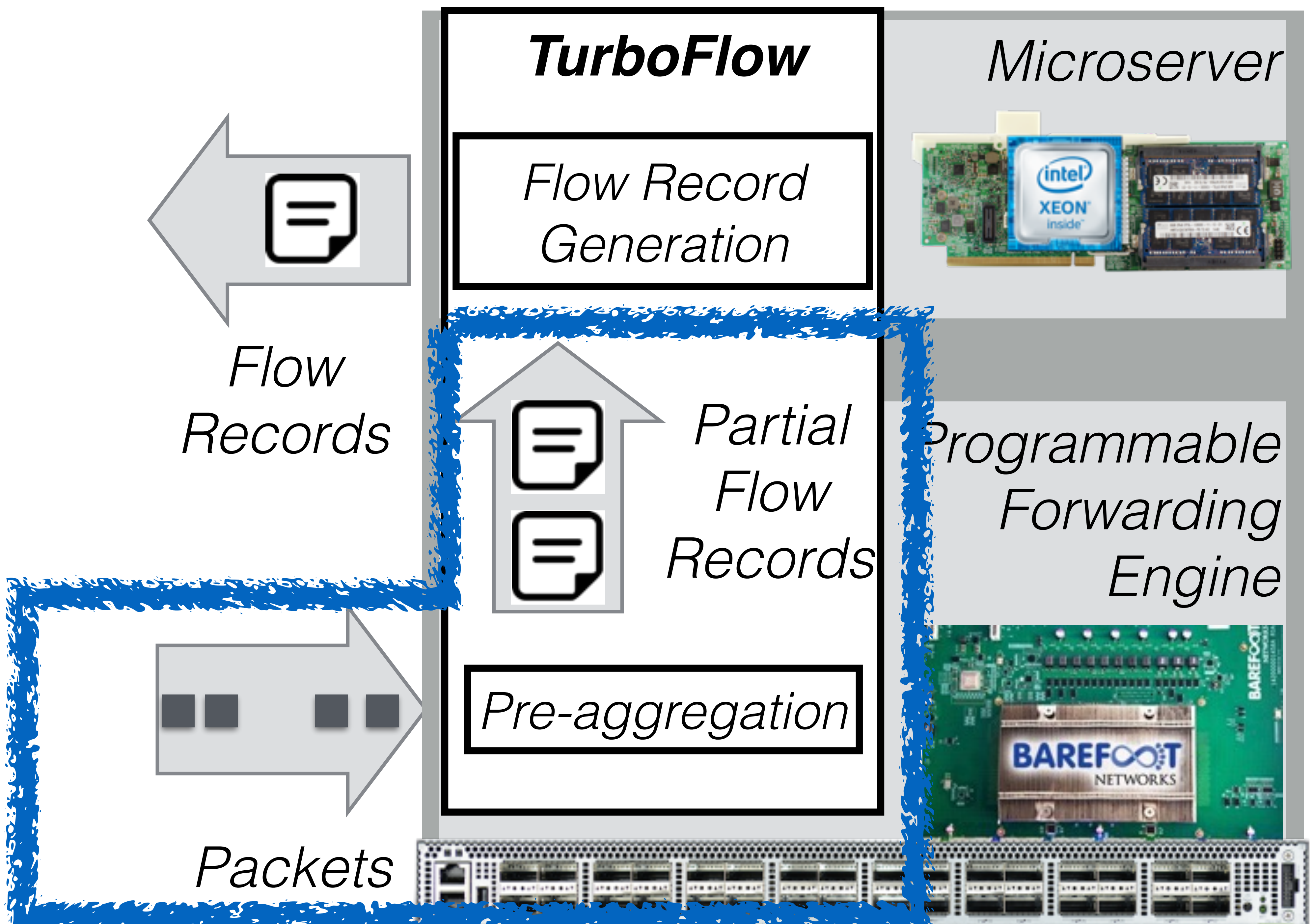


Outline

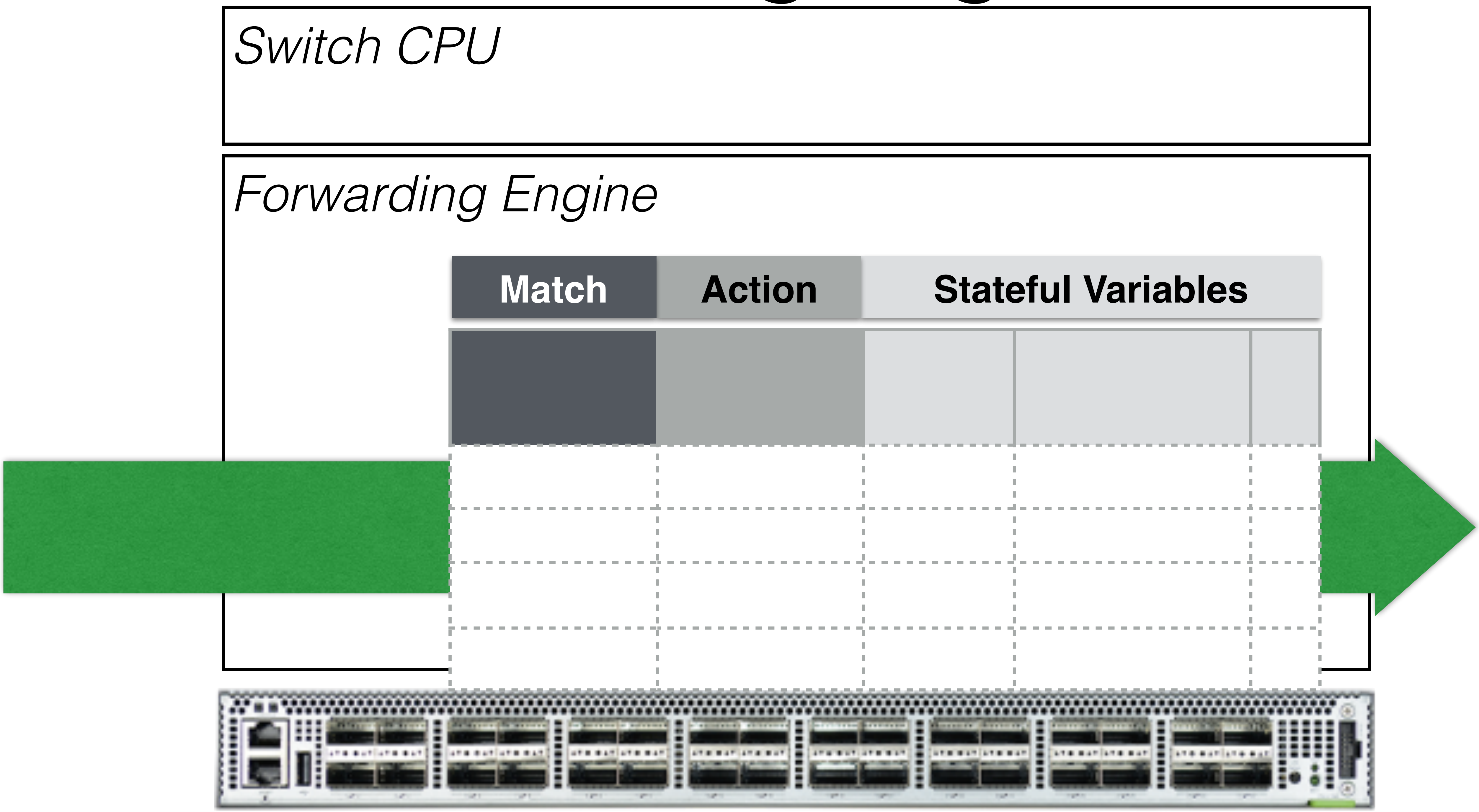
- Introduction
- **Architecture**
- Evaluation
- Conclusion



TurboFlow Architecture



Background: Programmable Forwarding Engines



Background: Programmable Forwarding Engines

Switch CPU

Forwarding Engine

Match	Action	Stateful Variables		
Flow (IP 5-tuple)		**Packet Count**	**Average Interarrival**	**…**
A -> B	Update	3	1 ms	…
E -> G	Update	49	8 ms	…
F -> G	Update	3	42 ms	…

Background: Programmable Forwarding Engines

Switch CPU

Table Manager

Rule installation rate: < 10 K / s

Forwarding Engine

Match	Action	Stateful Variables		
Flow (IP 5-tuple)		Packet Count	Average Interarrival	…
A -> B	Update	3	1 ms	…
E -> G	Update	49	8 ms	…
F -> G	Update	3	42 ms	…

Flow arrival rate @ 1 Tb/s: > 10,000 K / s

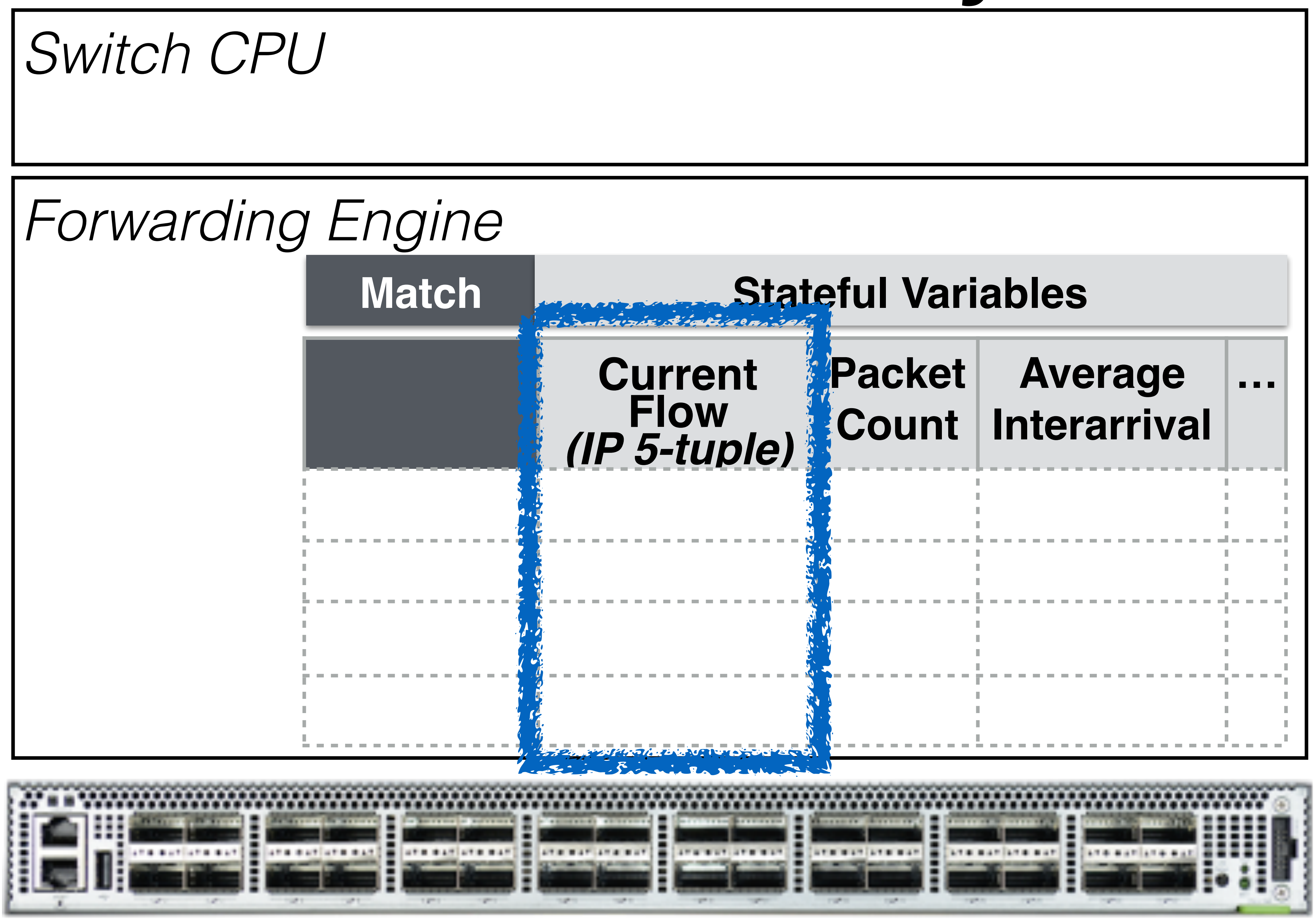
TurboFlow Architecture: Using the FE Efficiently

Switch CPU

~~Table Manager~~

Forwarding Engine

TurboFlow Architecture: Using the FE Efficiently



TurboFlow Architecture: Using the FE Efficiently

Switch CPU

Table Manager

Forwarding Eng

Match	Stateful Variables			
Flow Key Hash	**Current Flow *(IP 5-tuple)***	**Packet Count**	**Average Interarrival**	**…**
1				
2				
3				
4				

TurboFlow Architecture: Using the FE Efficiently

Switch CPU

Forwarding Engine

Tracked Flow:
Update Counters

A->B

HASH

Match	Stateful Variables			
Flow Key Hash	**Current Flow (IP 5-tuple)**	**Packet Count**	**Average Interarrival**	**…**
1	A -> B	**4**	**3 ms**	…
2	C -> D	49	8 ms	…
3	E -> F	3	42 ms	…
4	Z -> Q	9	10 ms	…

TurboFlow Architecture: Using the FE Efficiently

Switch CPU

Record Aggregator

Forwarding Engine

Tracked Flow:
Update Counters

HASH

A->B

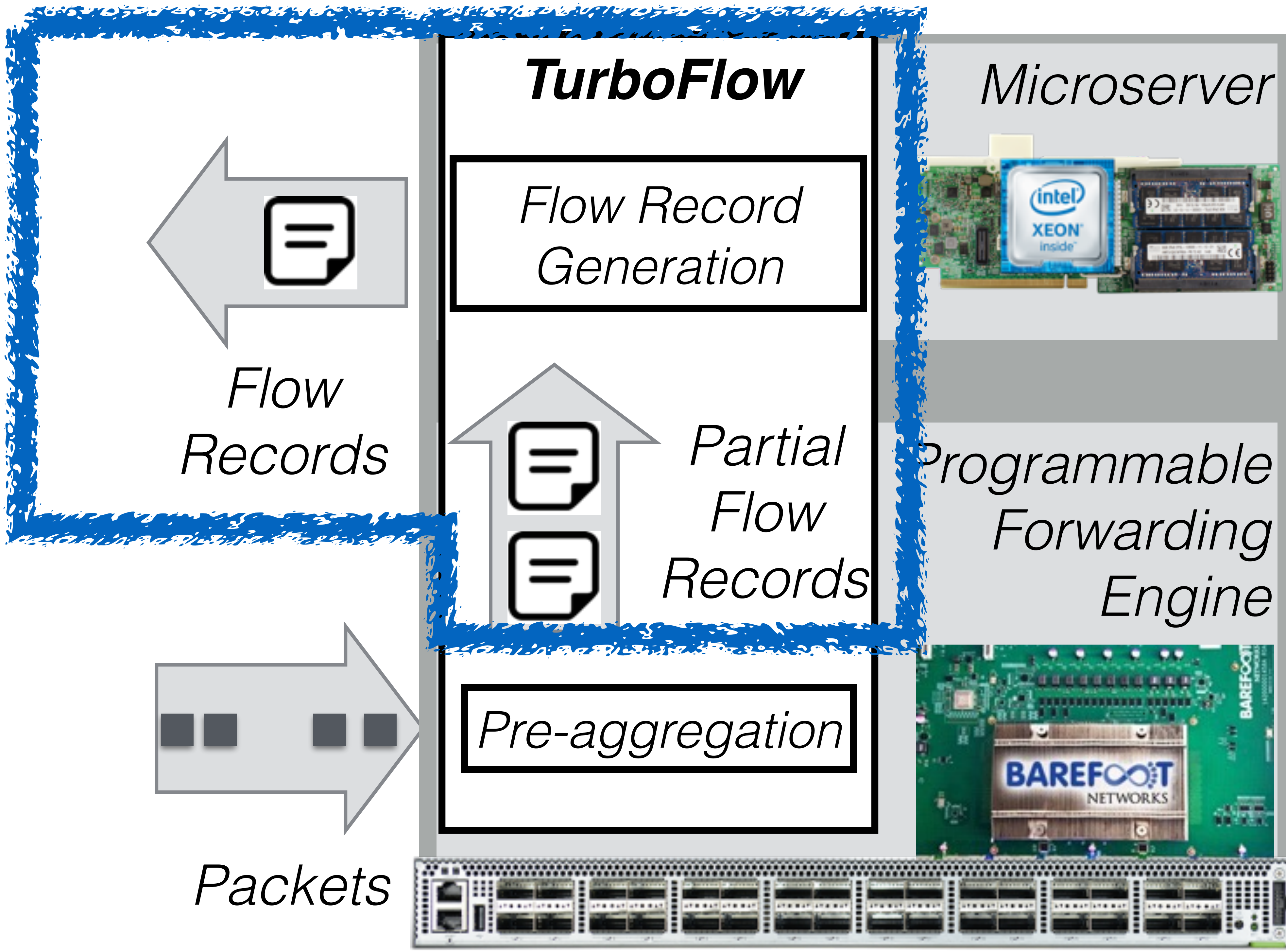
G->H

Match	Stateful Variables			
Flow Key Hash	**Tracked Flow *(IP 5-tuple)***	**Packet Count**	**Average Interarrival**	**…**
1	A -> B	4	3 ms	…
2	C -> D	49	8 ms	…
3	E -> F	3	42 ms	…
4	**G -> H**	**1**	**—**	…

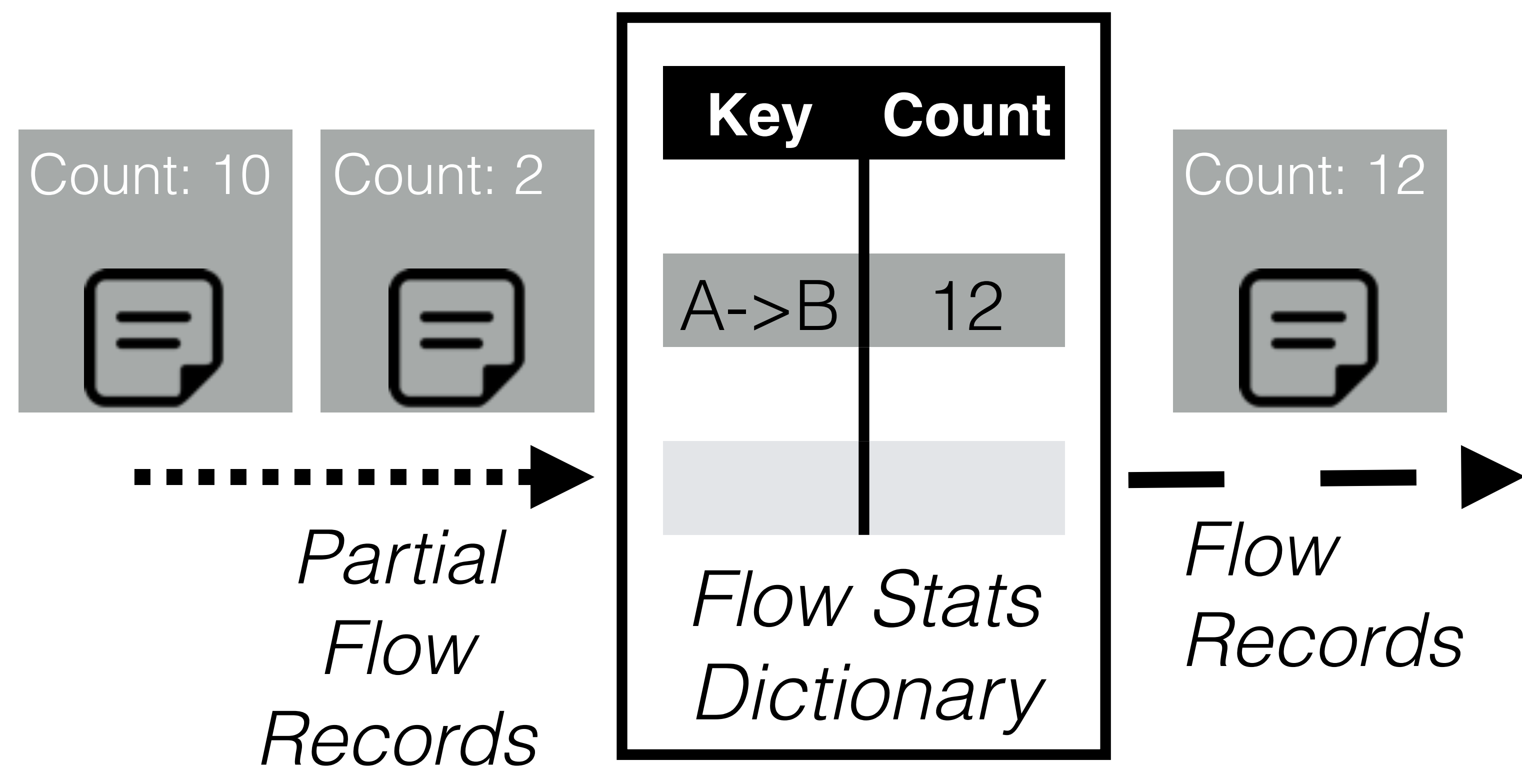
Z -> Q:
9
10 ms
…

Untracked Flow: Replace colliding record, send it to CPU

TurboFlow Design

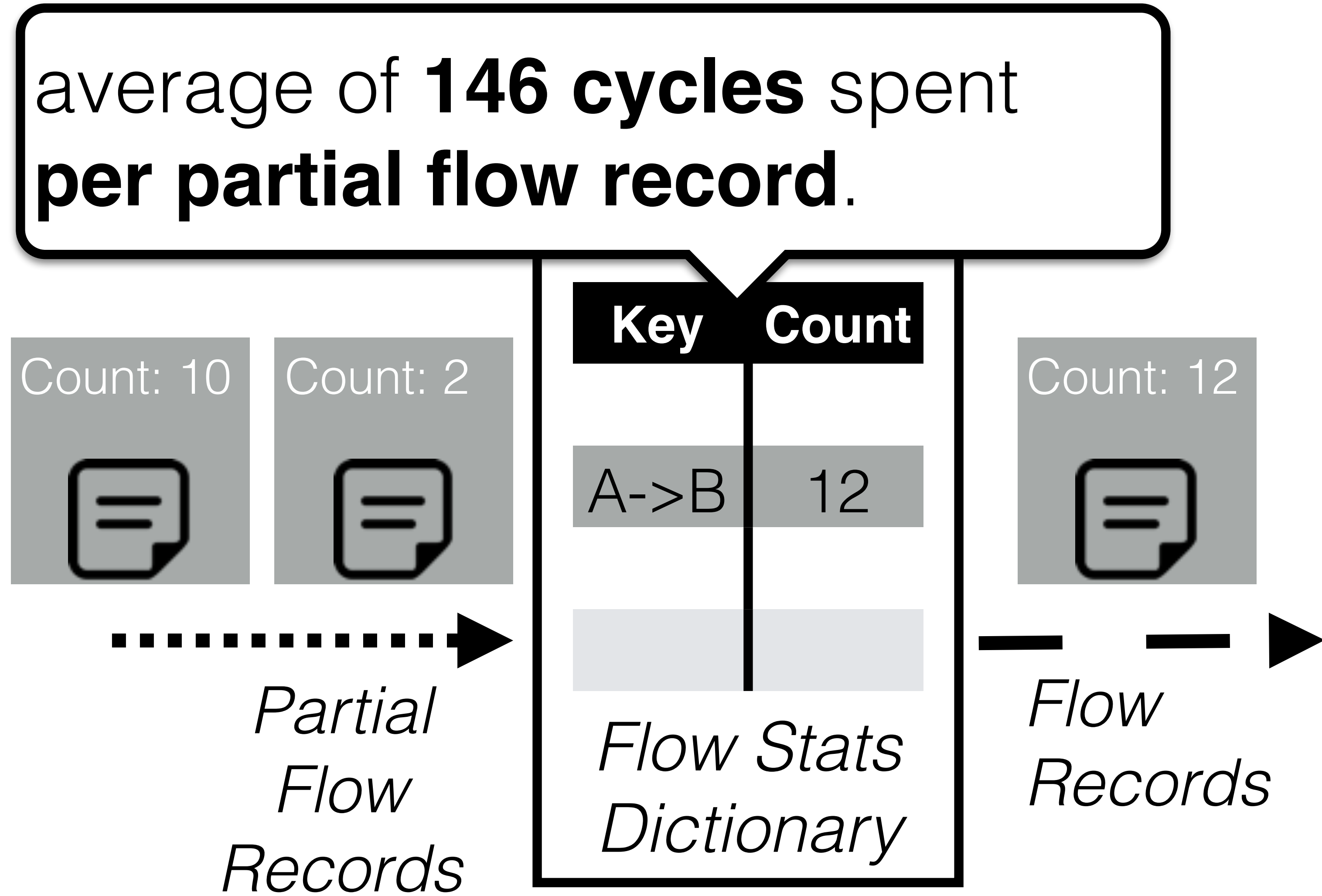


TurboFlow Architecture: Using the CPU Efficiently



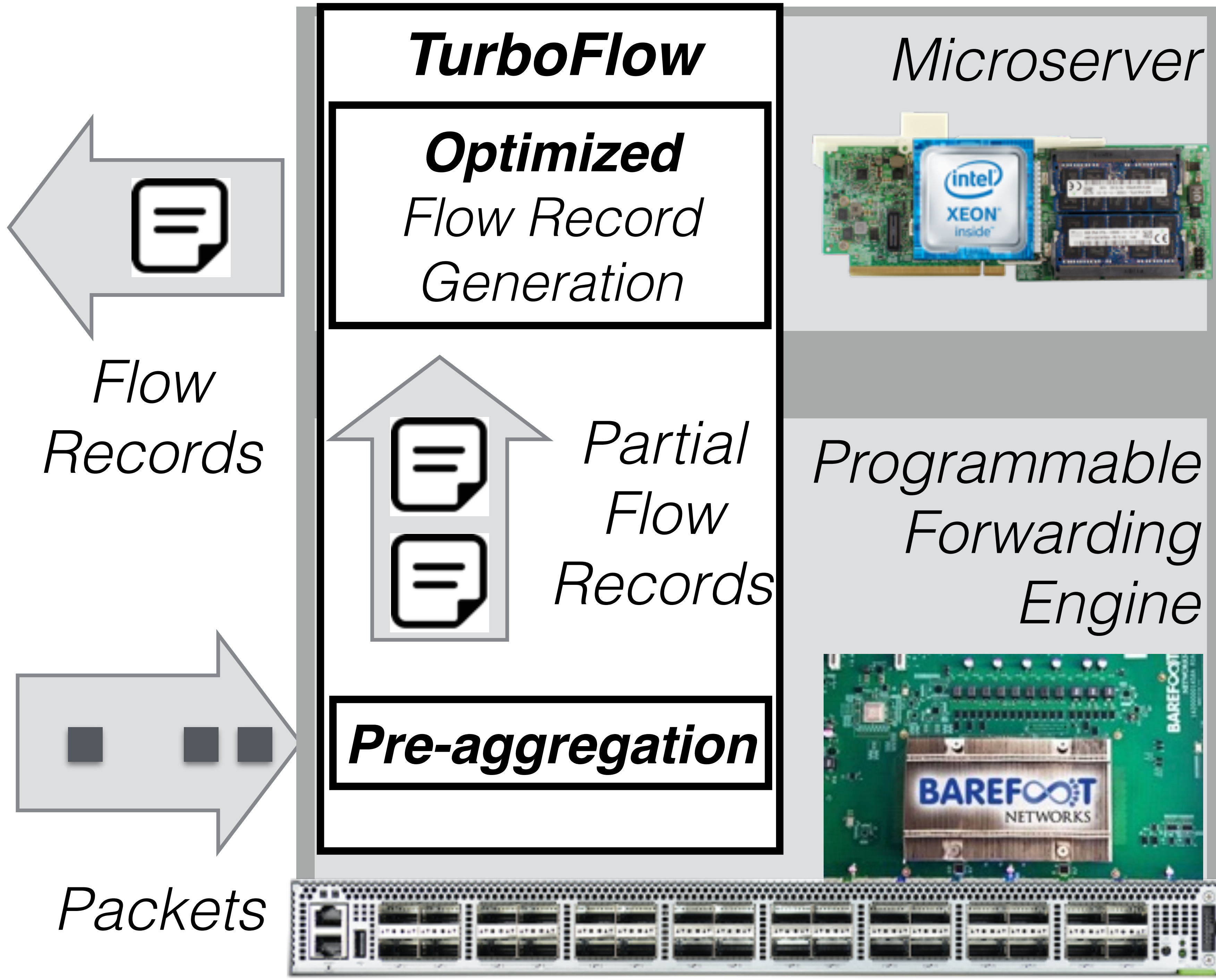
TurboFlow Architecture: Using the CPU Efficiently

Optimization	Performance Vs. Baseline
baseline *(std::unordered_map)*	-
Reduce Pointer Operations	**1.64X**
Vectorize Key Comparison	**3.79X**
Batch and Prefetch	**4.9X**



Outline

- Introduction
- Architecture
- **Evaluation**
- Conclusion



Implementation and Evaluation

Implementations

P4 Switch

(3.2 Tb/s Barefoot Tofino)

P4 SmartNIC

(40 Gb/s Netronome NFP)

Benchmark Workloads

- **10 Gb/s Internet Router Traces** *(CAIDA 2015)*

- **144 Node Simulated Datacenter Cluster** *(YAPS simulator)*

Implementation and Evaluation

Implementations

P4 Switch

(3.2 Tb/s Barefoot Tofino)

P4 SmartNIC

(40 Gb/s Netronome NFP)

Benchmark Workloads

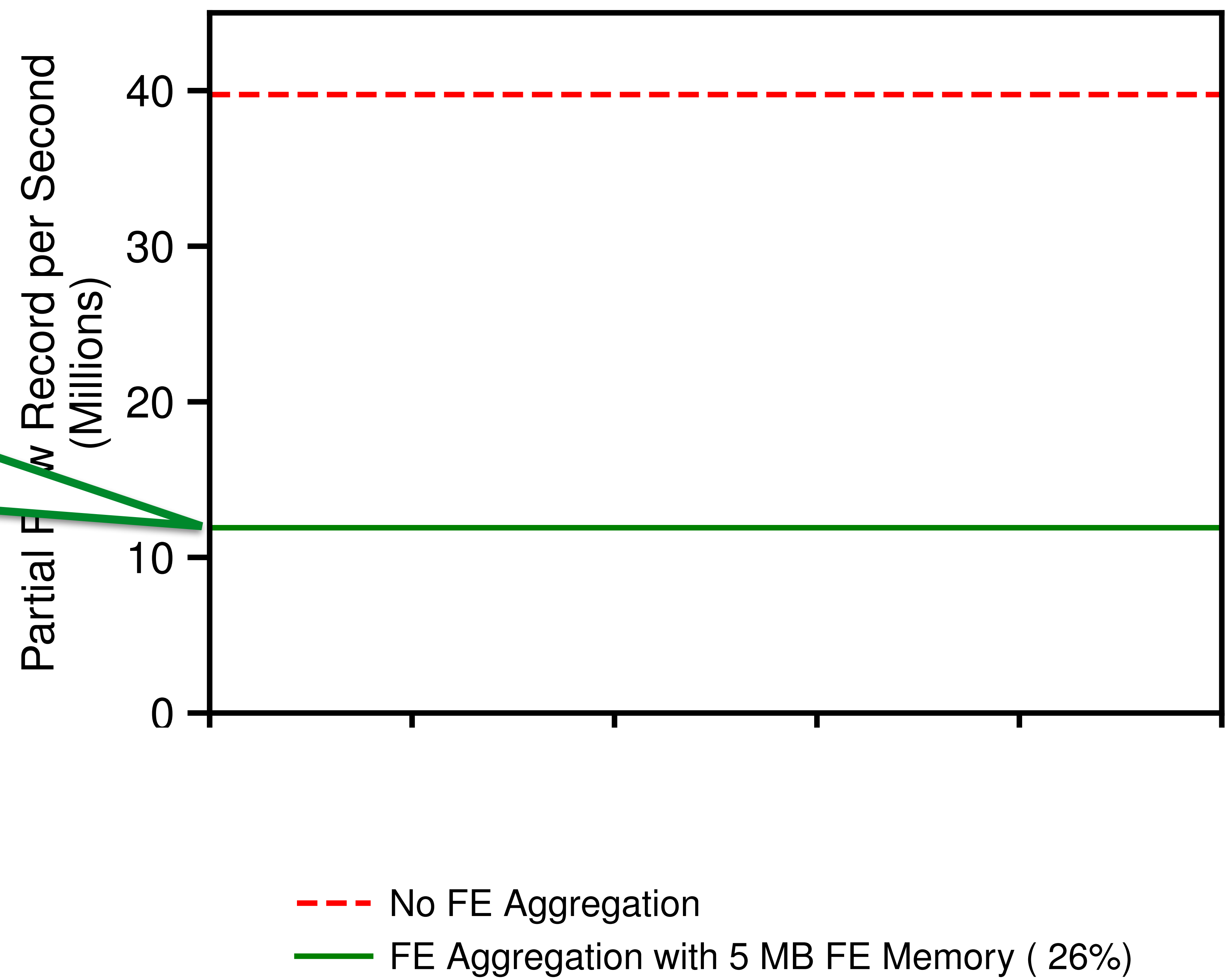
- **10 Gb/s Internet Router Traces** *(CAIDA 2015)*
- **144 Node Simulated Datacenter Cluster** *(YAPS simulator)*

Required Average Throughput to Monitor 100 X 10 Gb/s Internet Links

Partial Flow Record per Second
(Millions)

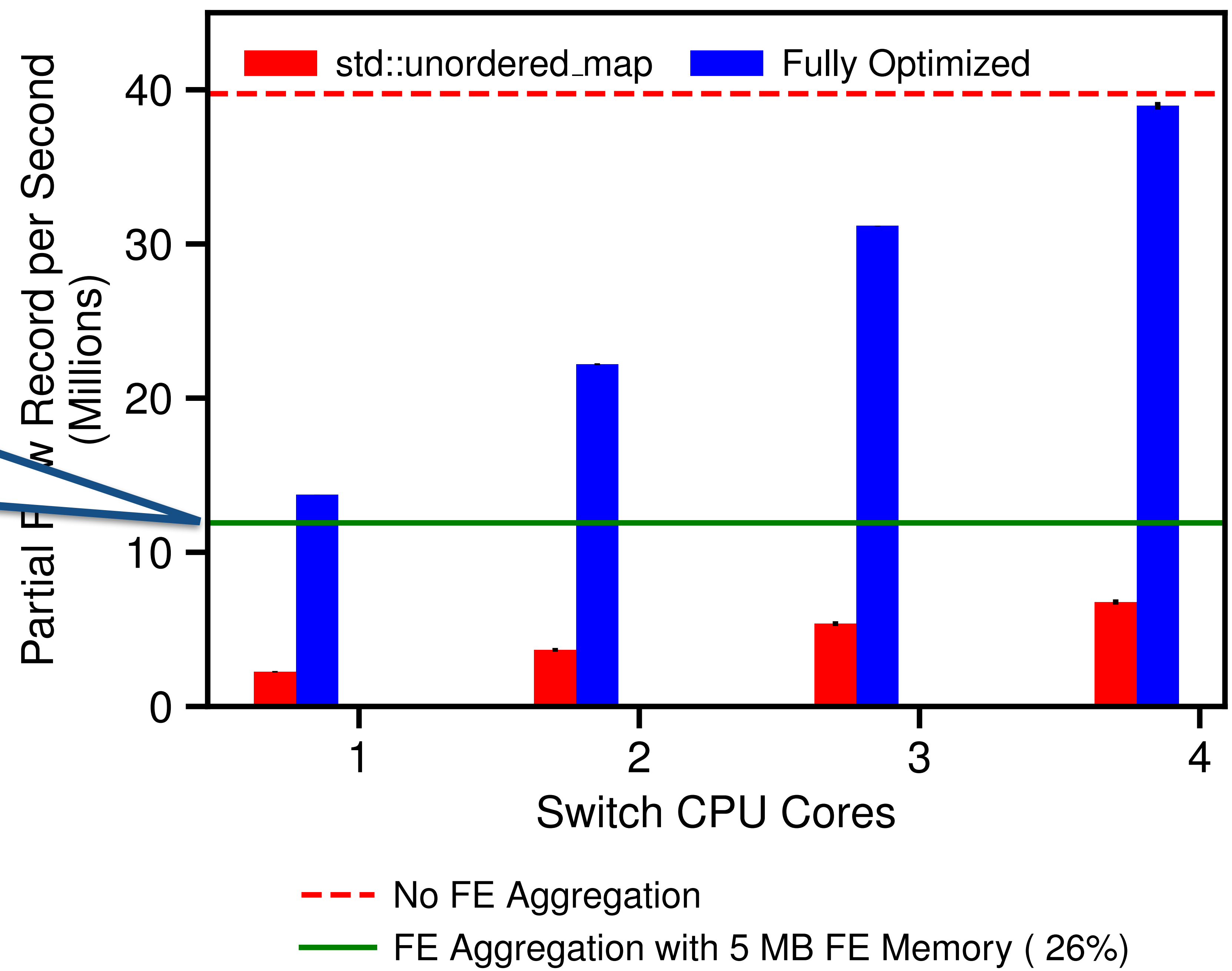
Required Average Throughput to Monitor 100 X 10 Gb/s Internet Links

Partial aggregation using 5 MB of FE memory reduces workload by ~4X.

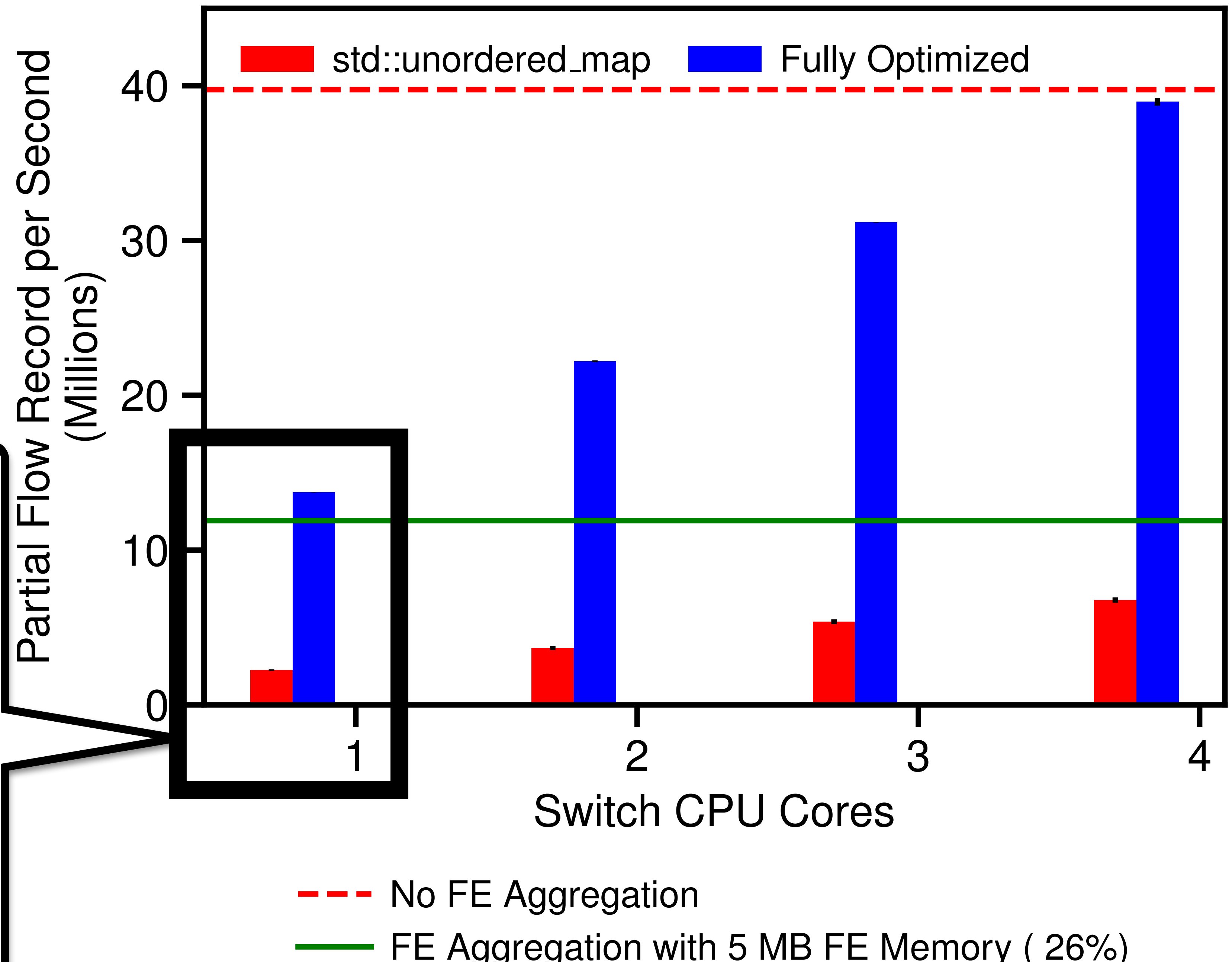


Required Average Throughput to Monitor 100 X 10 Gb/s Internet Links

Optimizations improve performance by ~5X.



Required Average Throughput to Monitor 100 X 10 Gb/s Internet Links

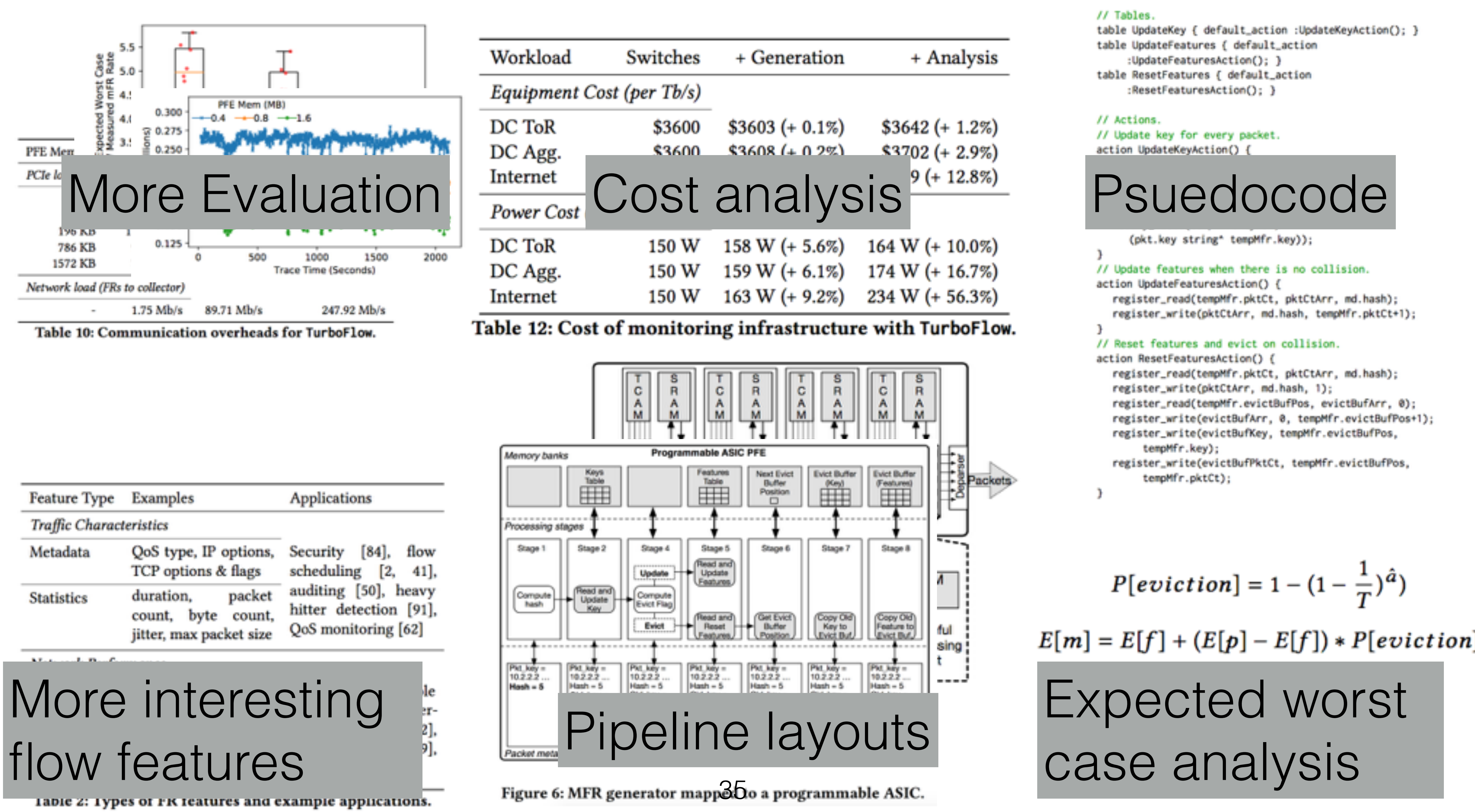


FE pre-aggregation + optimizations = terabit rate workloads using 1 core and ~26% of FE memory.

Outline

- Introduction
- TurboFlow Design
- Implementation and Evaluation
- **Conclusion**

In the Paper


More Evaluation
Cost analysis
Psuedocode
More interesting flow features
Pipeline layouts
Expected worst case analysis

Conclusion

(and Thank You for Listening!)

- Flow records are important for monitoring, but difficult to generate at the switch due to high traffic rates.
- **TurboFlow** is a flow record generator carefully optimized for next generation commodity switch hardware that scales to **multi-terabit rate traffic without sampling.**