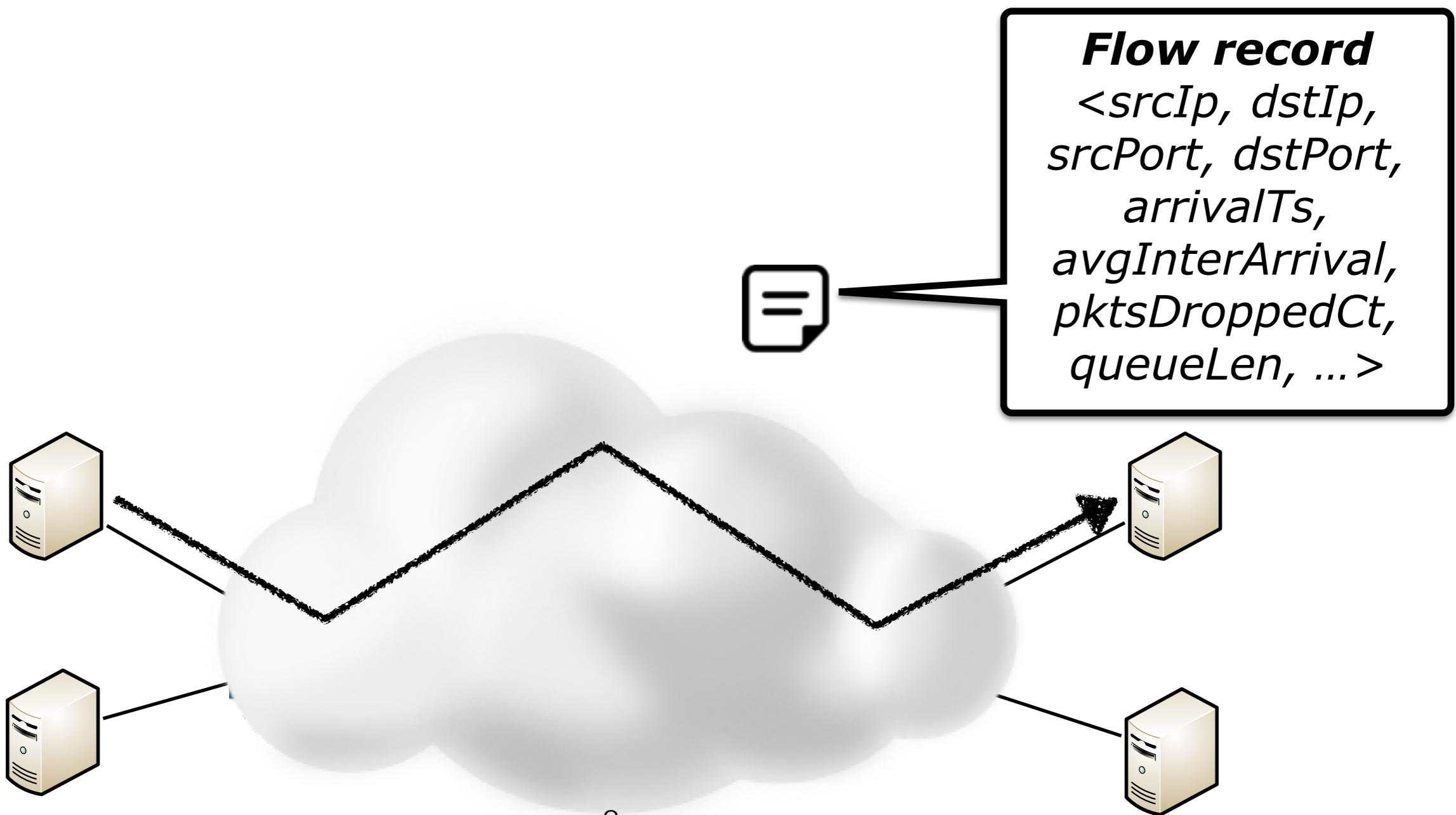


TurboFlow: Information Rich Flow Record Generation on Commodity Switches

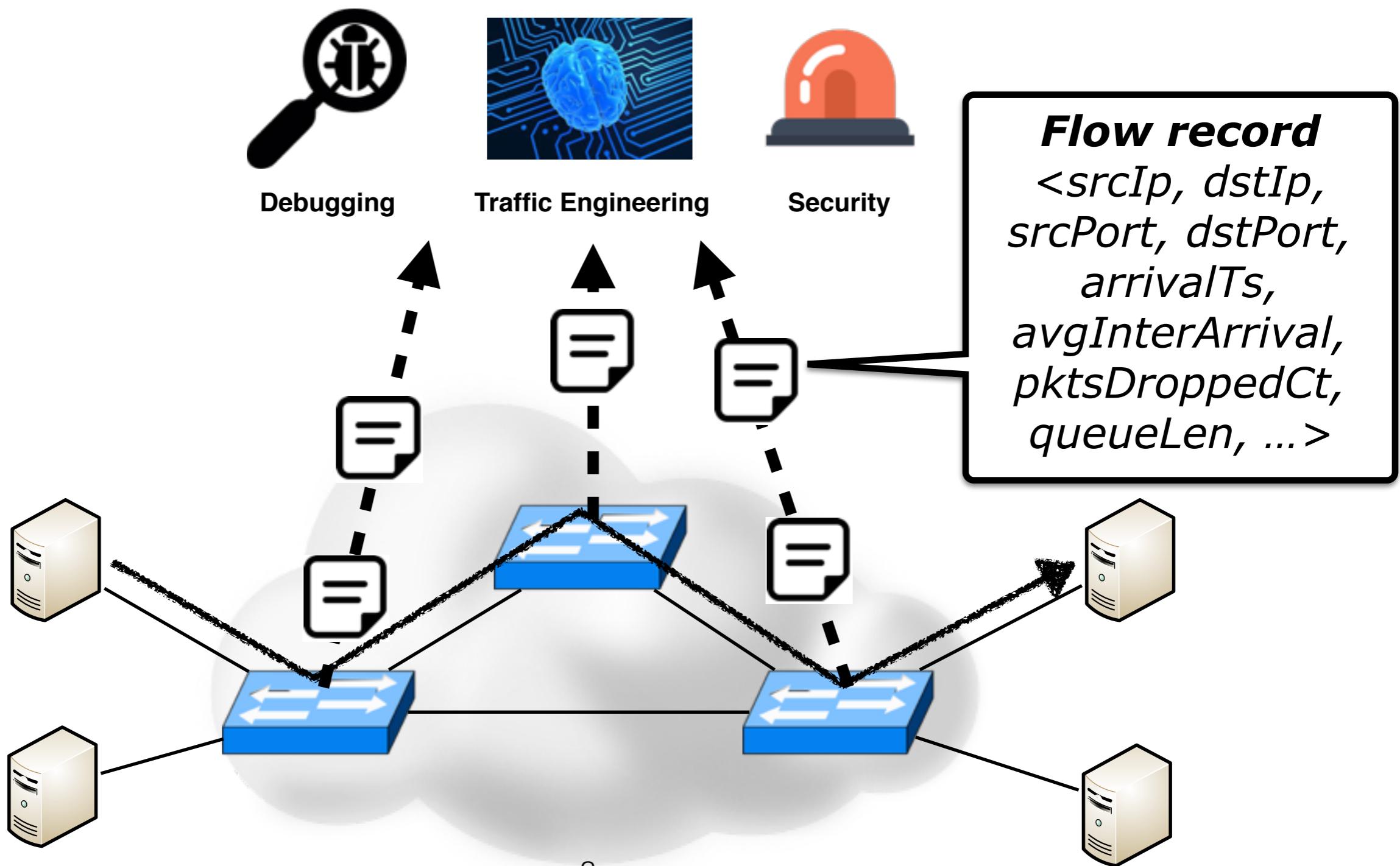
John Sonchack¹, Adam J. Aviv², Eric Keller³, Jonathan M. Smith¹

¹*University of Pennsylvania*, ²*USNA*, ³*University of Colorado*

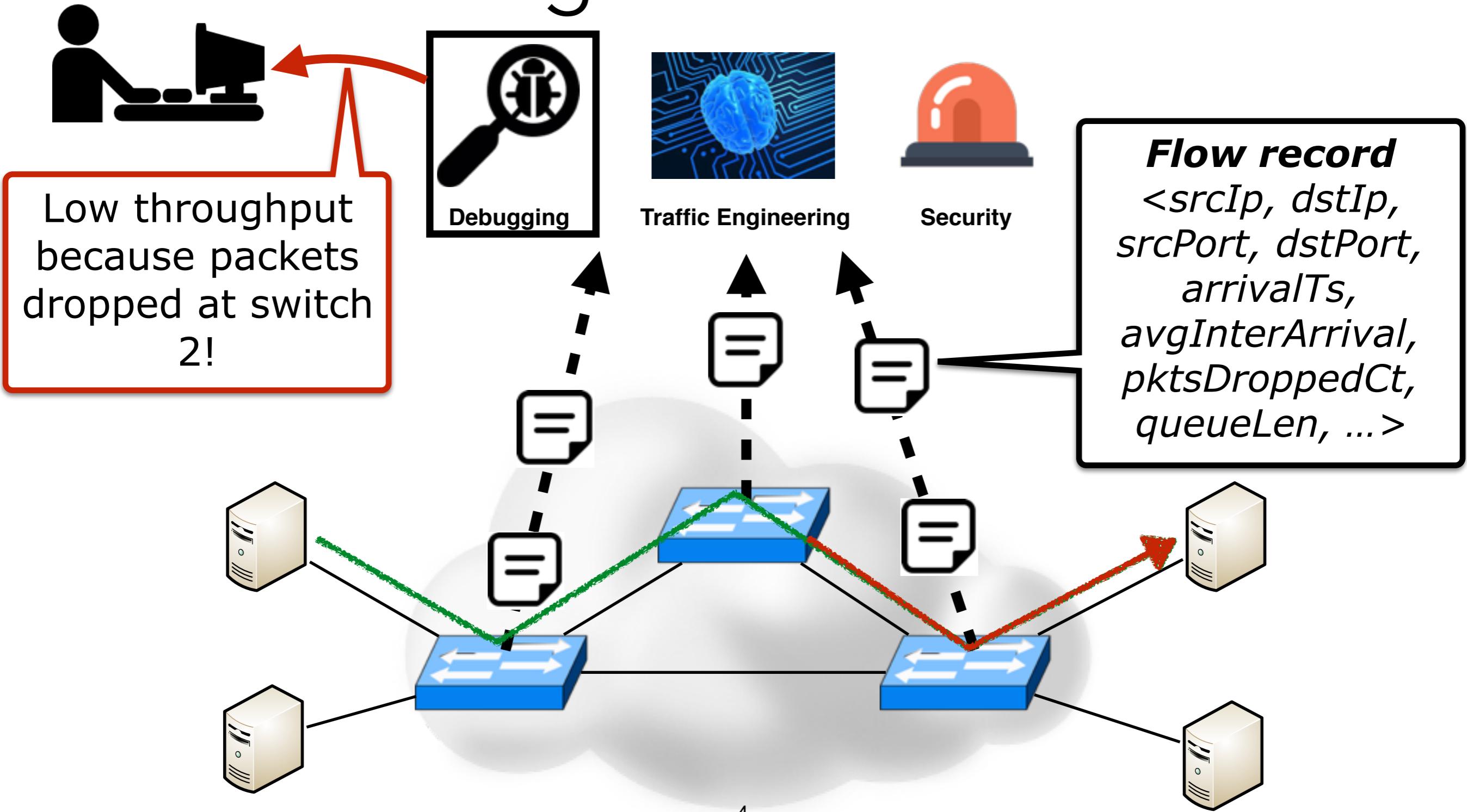
Introduction: Network Monitoring with Flow Records



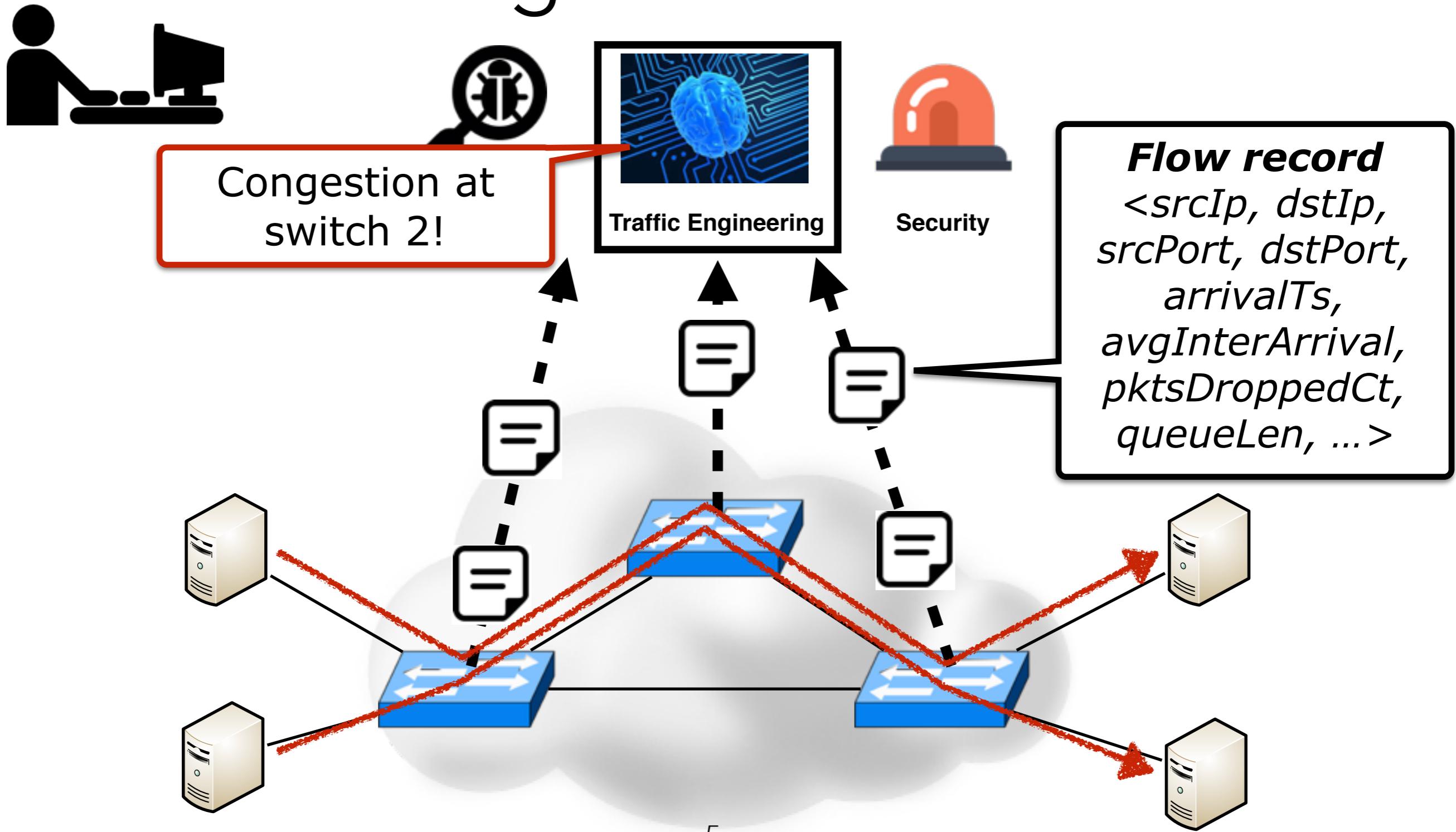
Introduction: Network Monitoring with Flow Records



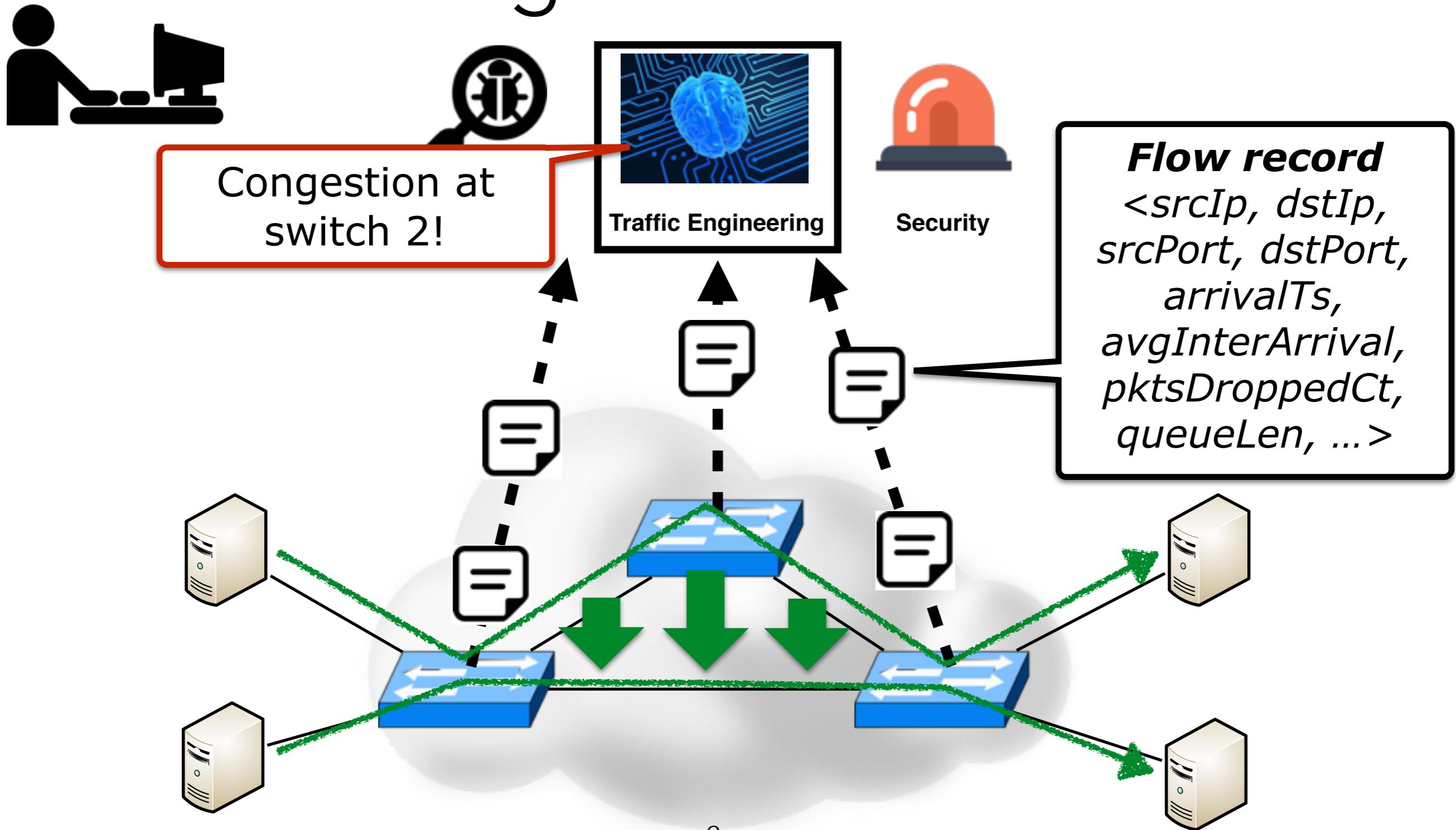
Introduction: Network Monitoring with Flow Records



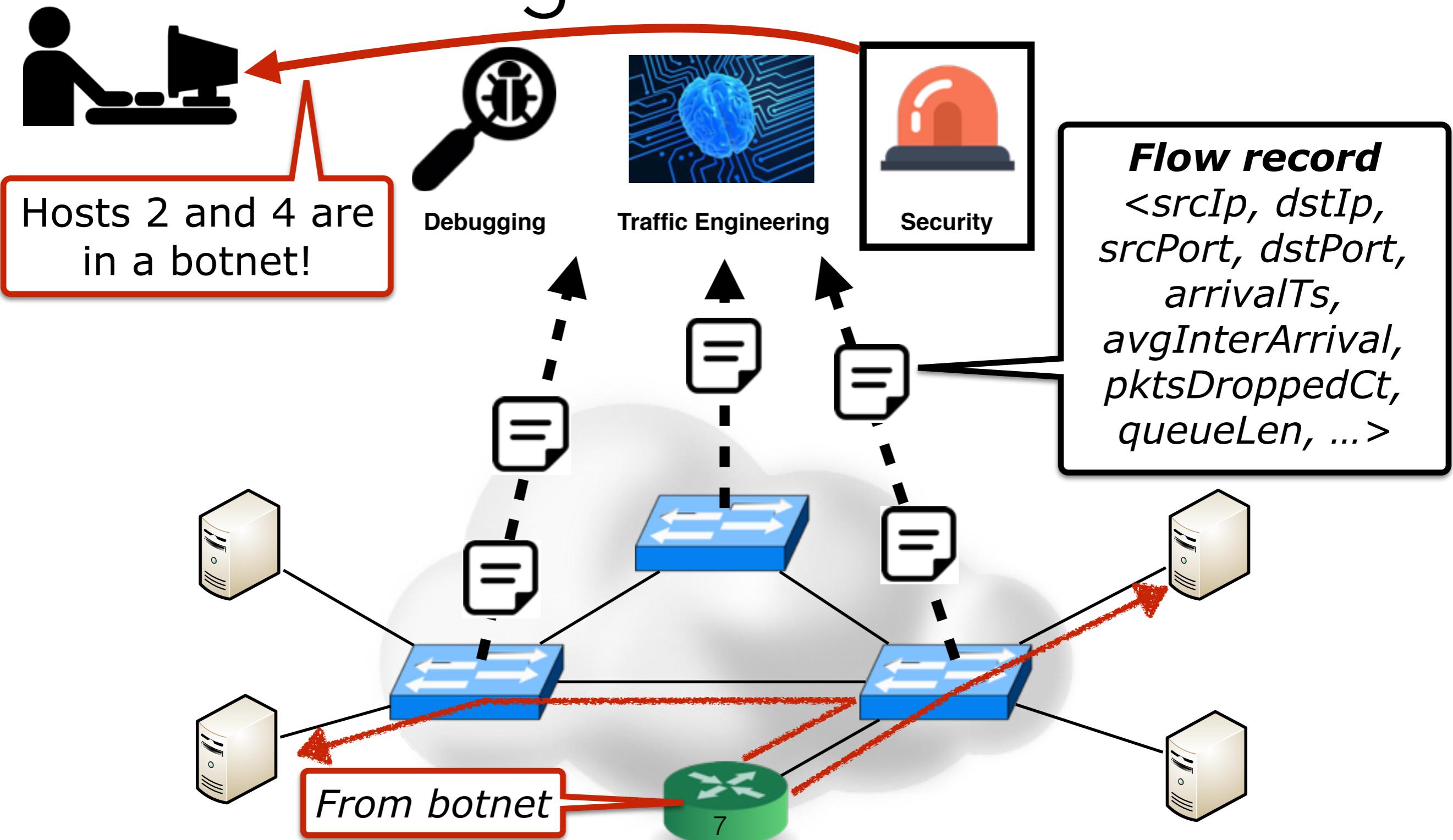
Introduction: Network Monitoring with Flow Records



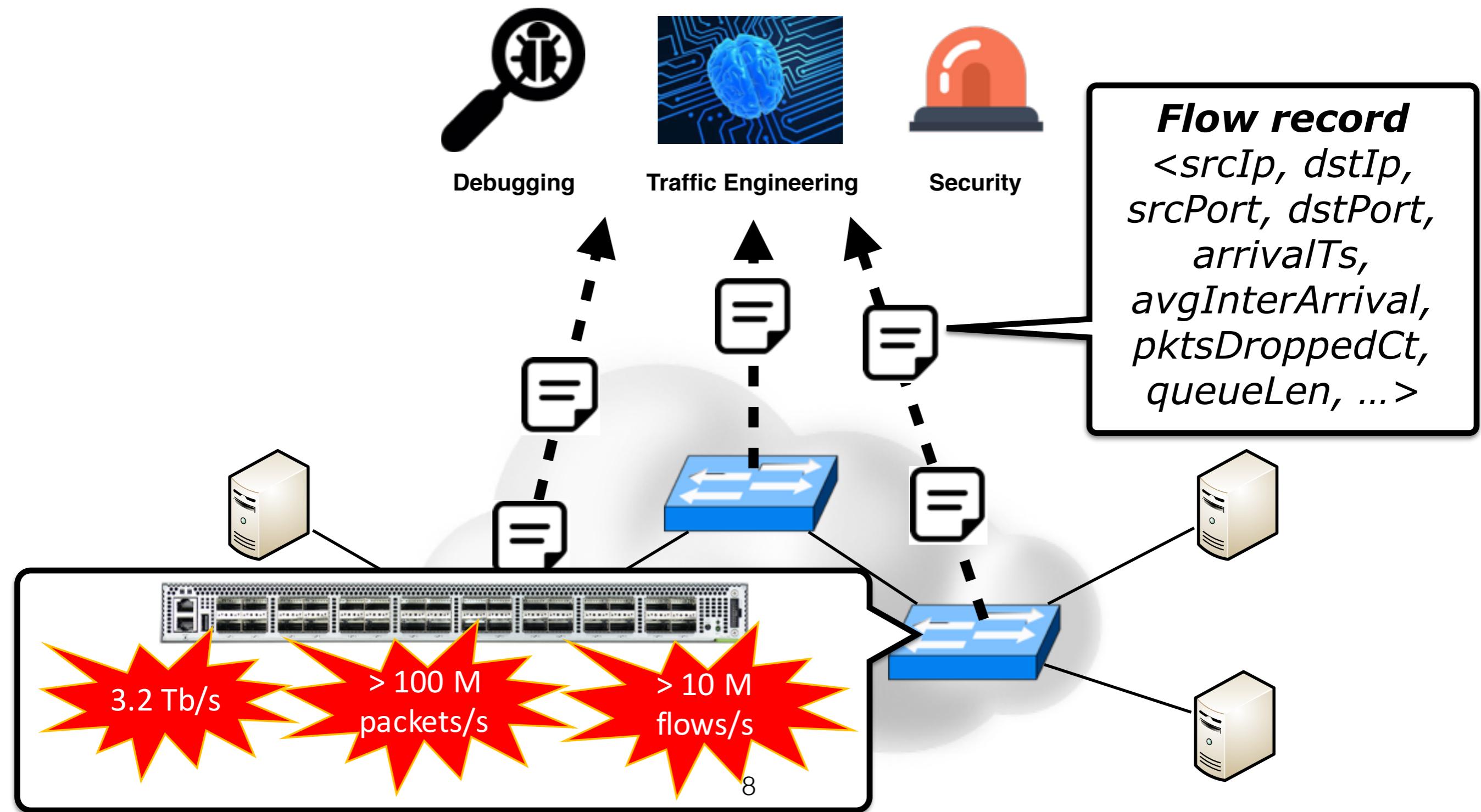
Introduction: Network Monitoring with Flow Records



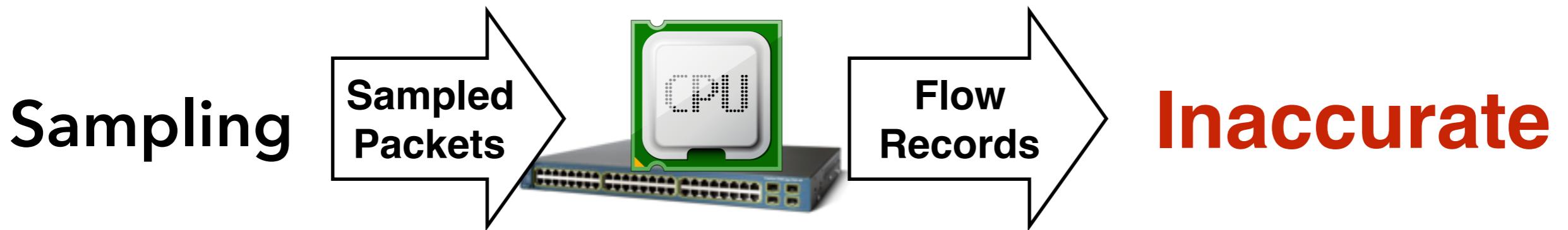
Introduction: Network Monitoring with Flow Records



Introduction: Network Monitoring with Flow Records



Flow Monitoring Switches: Prior Work



Flow Monitoring Switches:

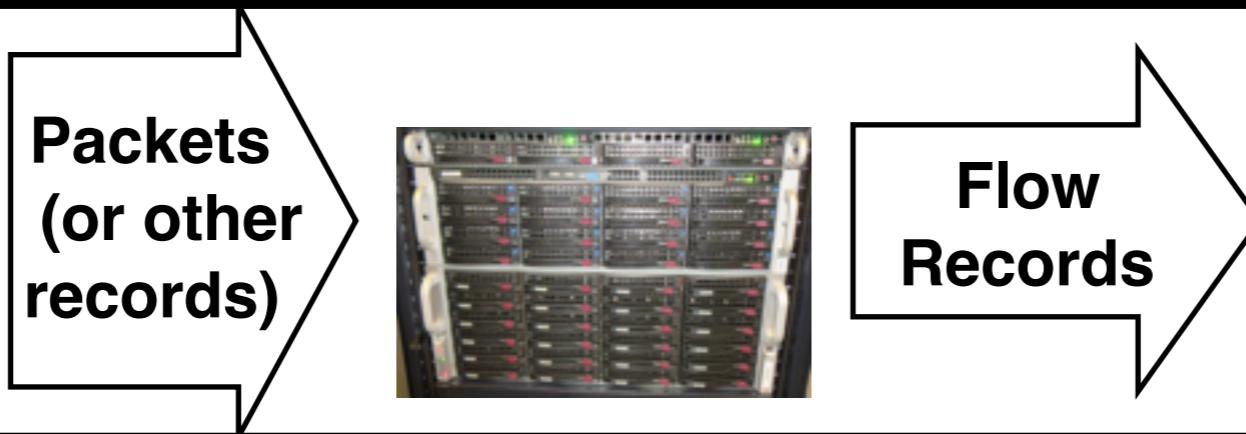
Prior Work

Sampling



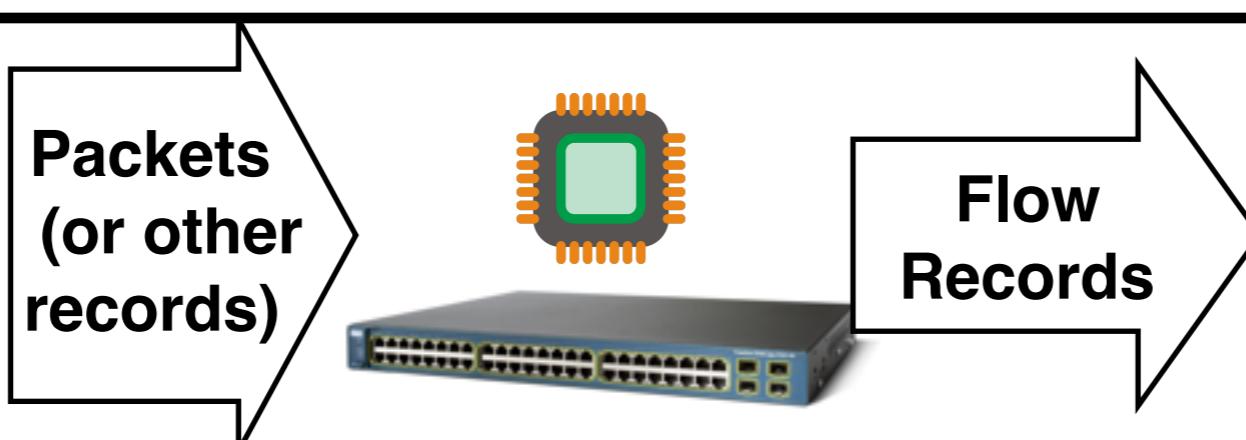
Inaccurate

Server Offloading



Expensive

Custom Hardware Offloading



Restrictive

Introduction: TurboFlow

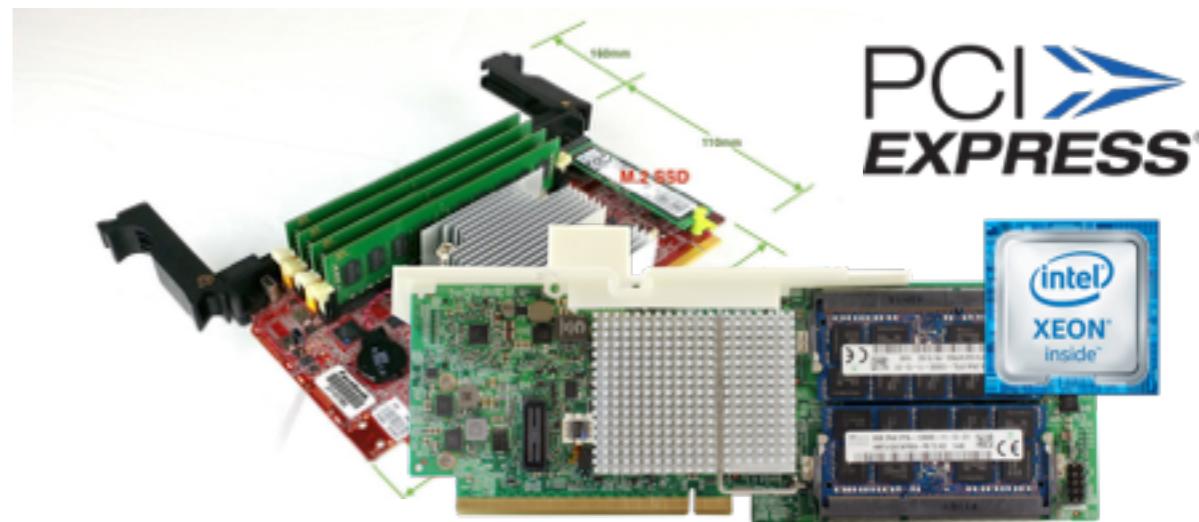
Main idea: Optimize *instead of offload*.

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

Programmable Forwarding Engines



Onboard Microservers



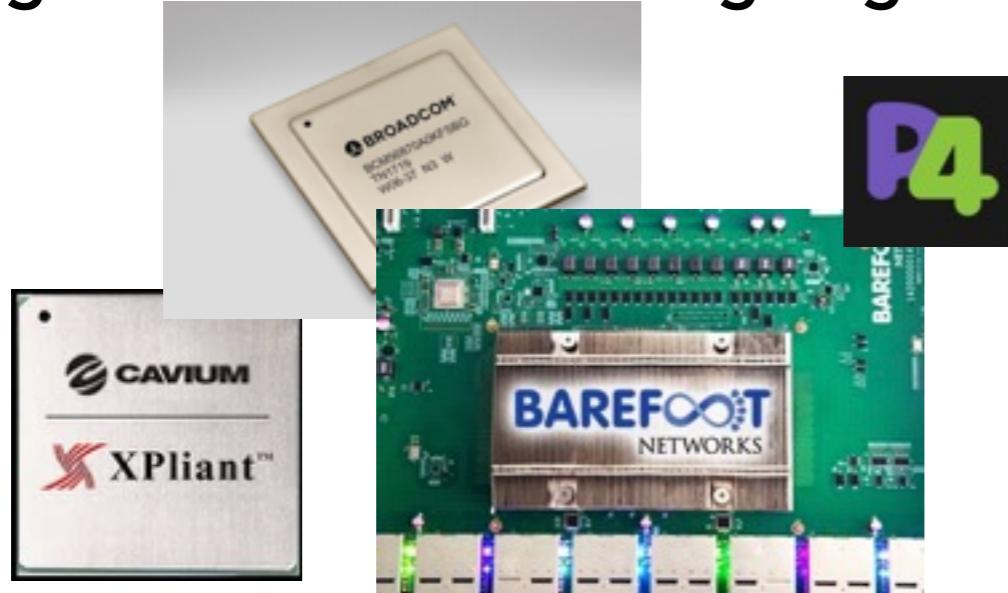
Introduction: TurboFlow

Main idea: Optimize *instead of offload*.

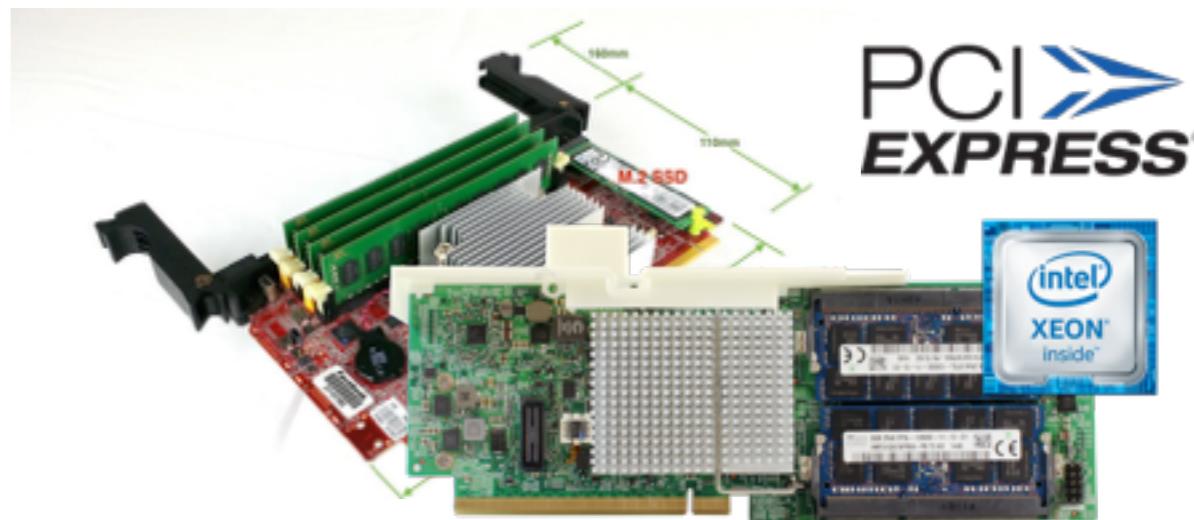
Q : What can we get out of the programmable hardware in next-generation commodity switches?

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

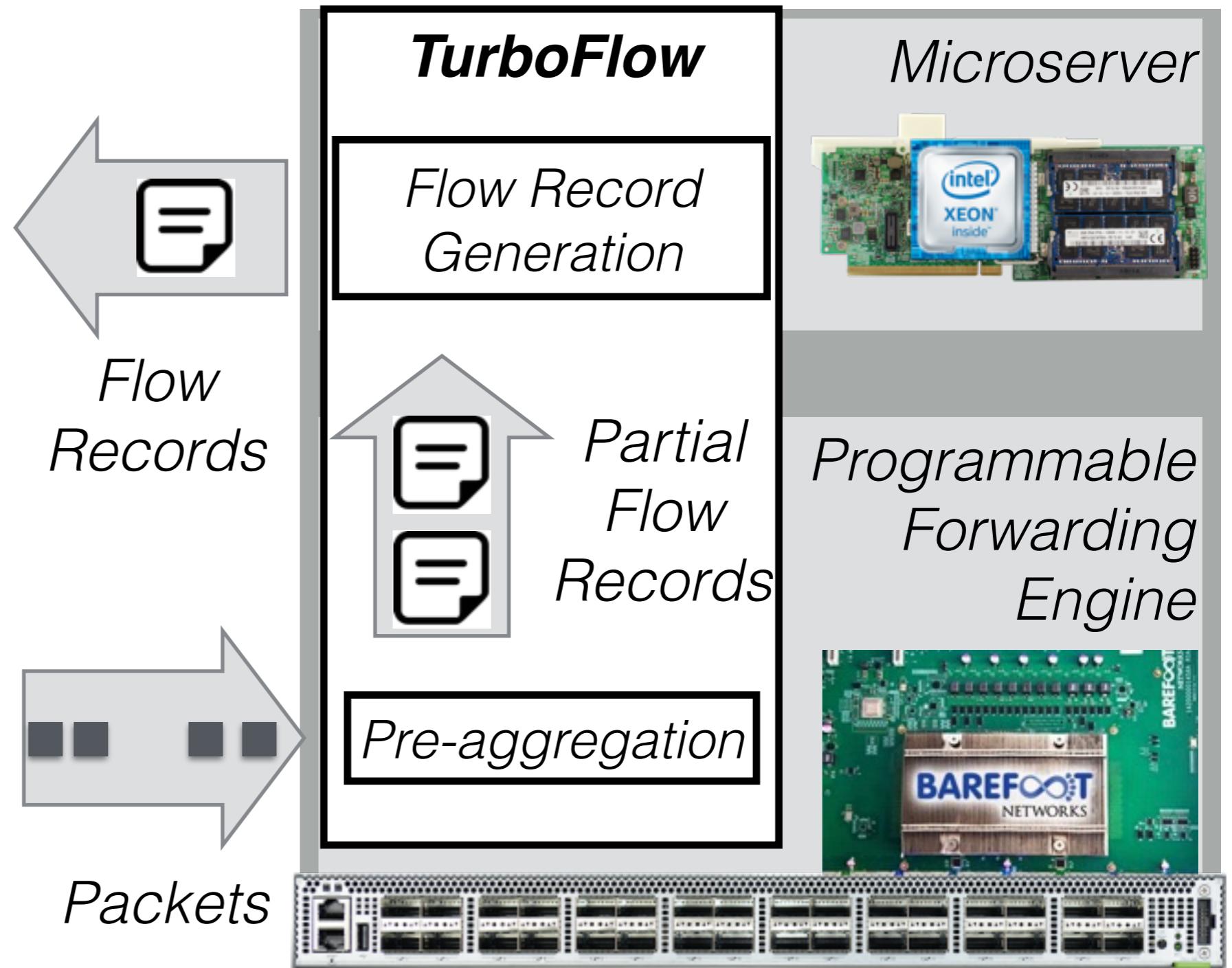
Programmable Forwarding Engines



Onboard Microservers

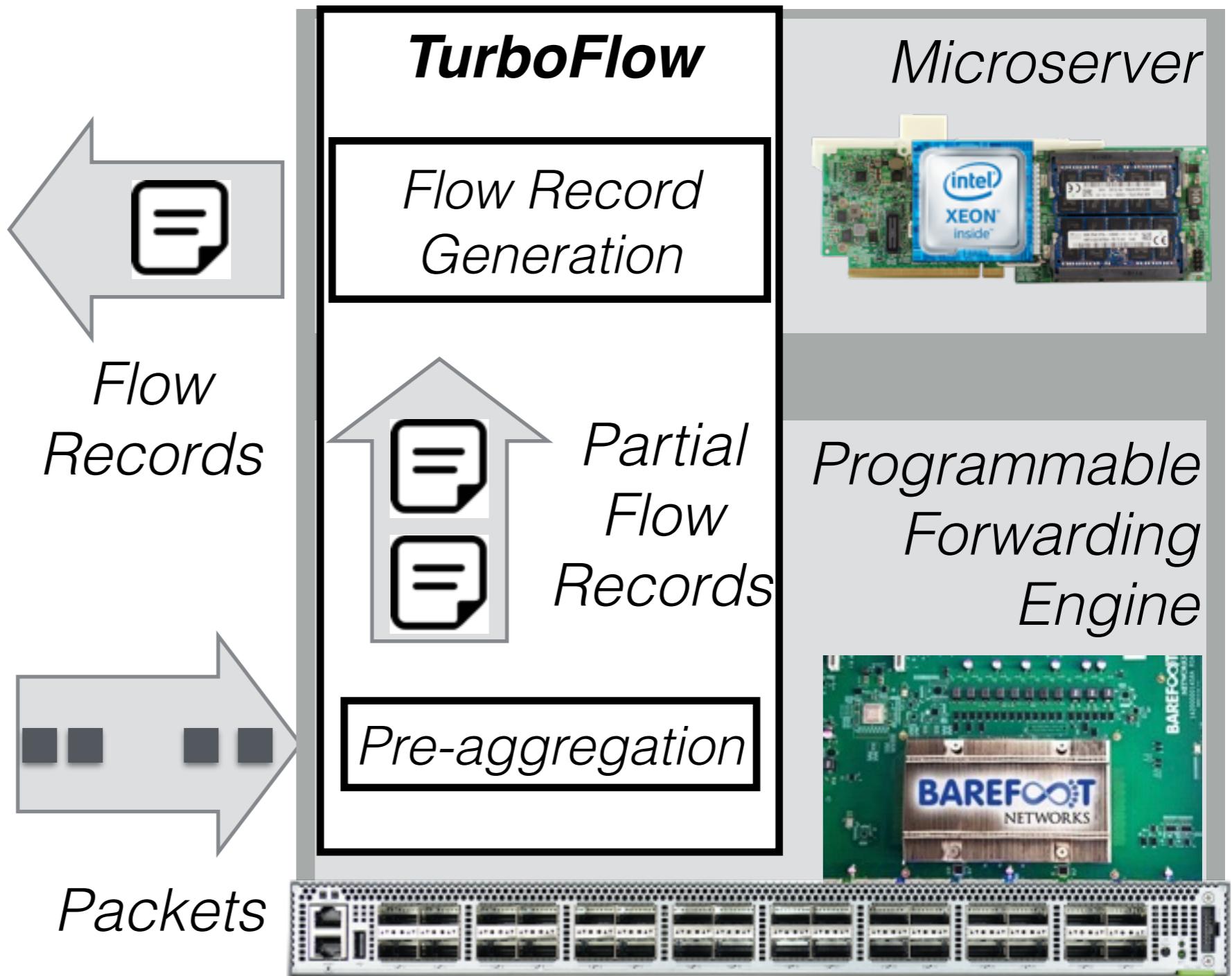


Introduction: TurboFlow

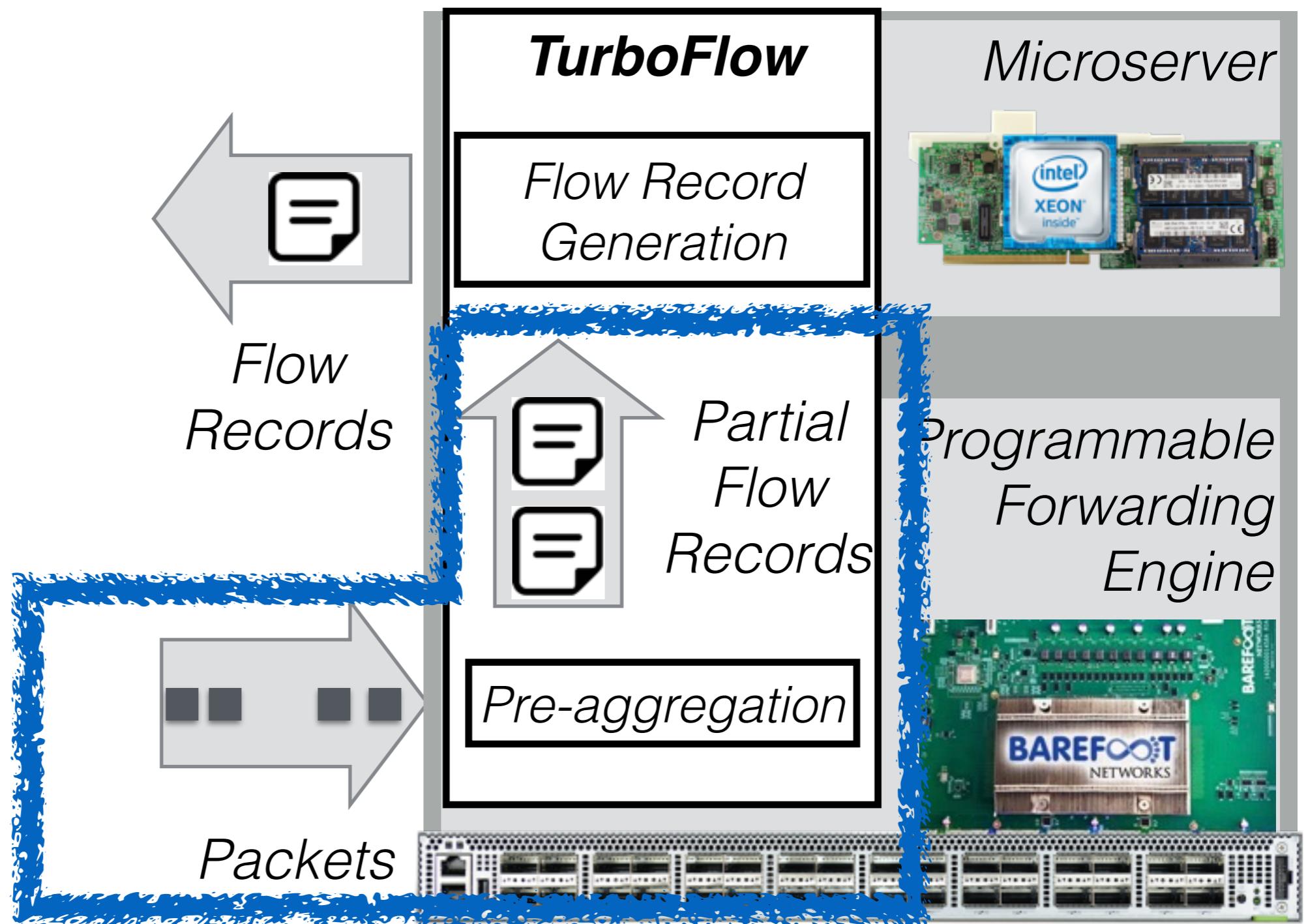


Outline

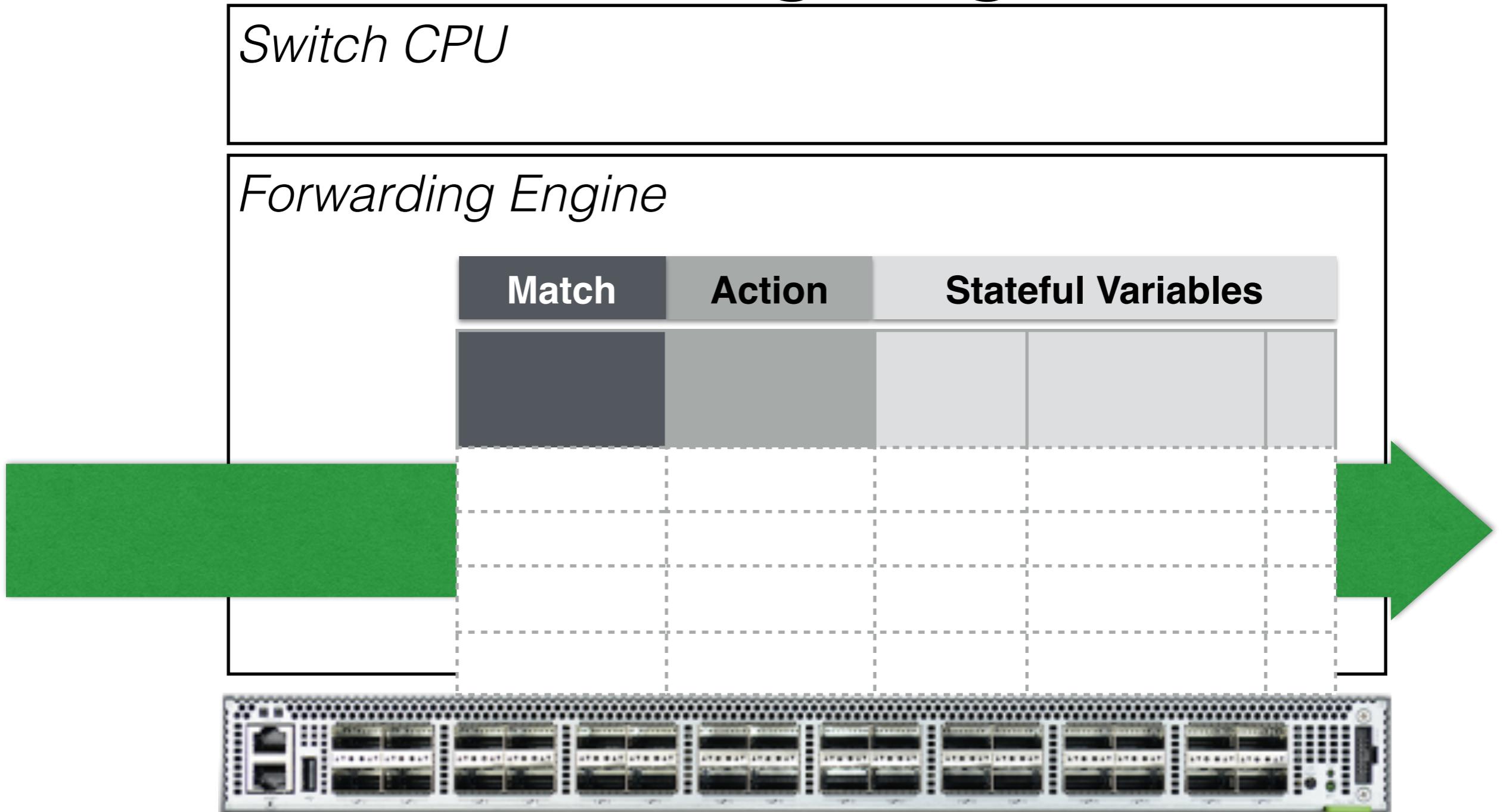
- Introduction
- **Architecture**
- Evaluation
- Conclusion



TurboFlow Architecture



Background: Programmable Forwarding Engines



Background: Programmable Forwarding Engines

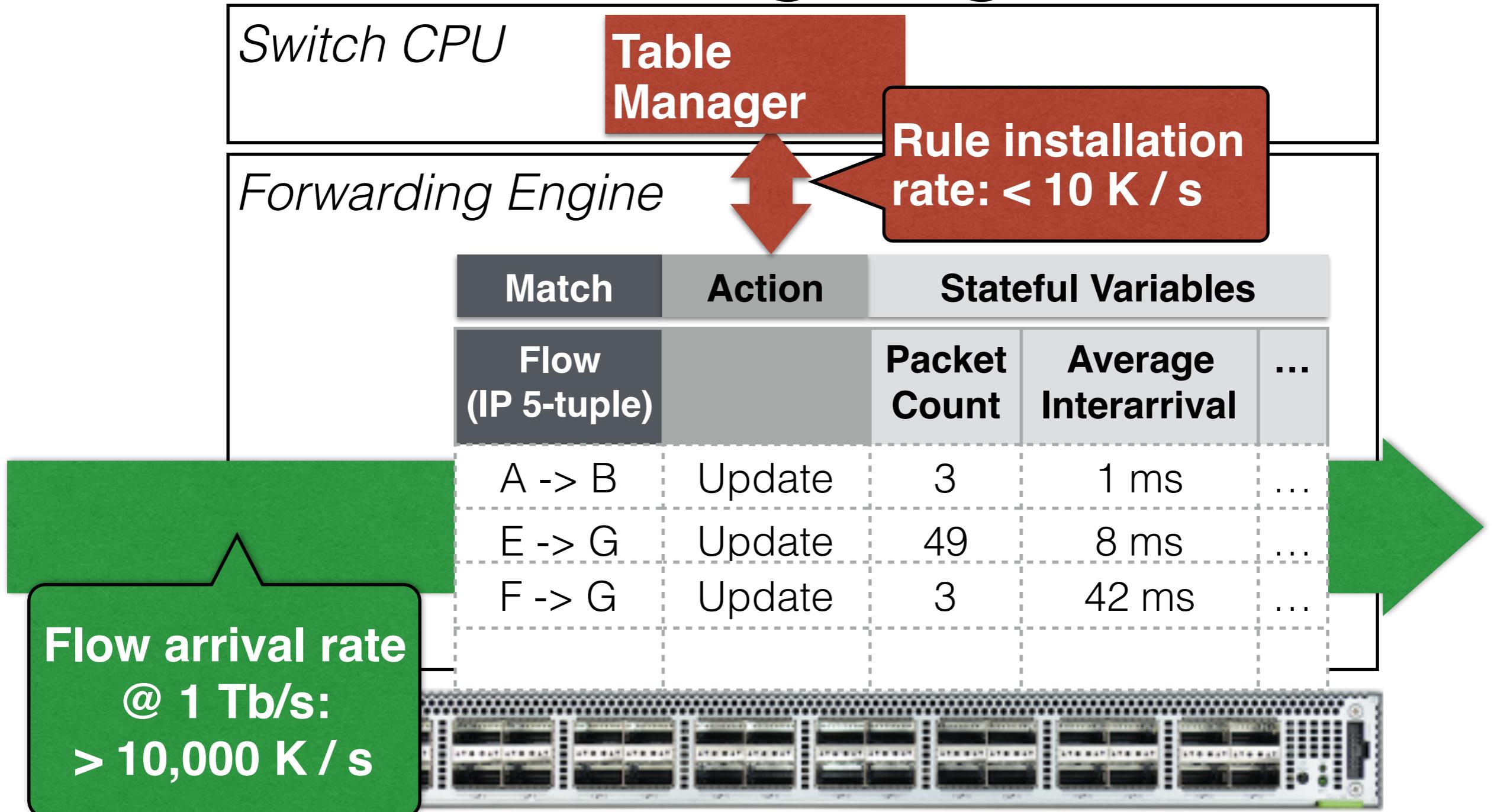
Switch CPU

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



TurboFlow Architecture: Using the FE Efficiently

Switch CPU

Table
Manager

Forwarding Engine



TurboFlow Architecture: Using the FE Efficiently

Switch CPU

Forwarding Engine

Match

Stateful Variables

Current
Flow
(IP 5-tuple)

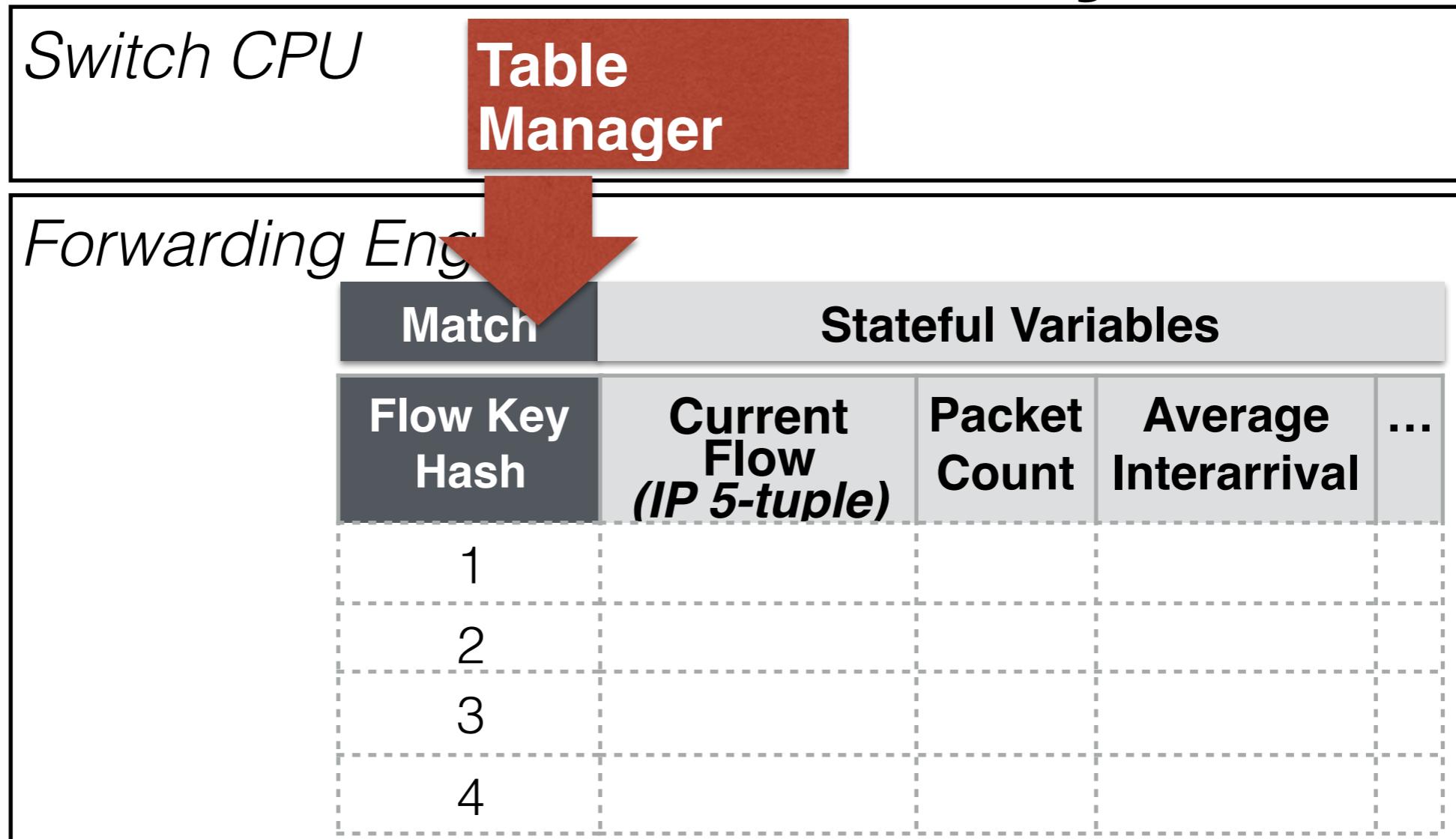
Packet
Count

Average
Interarrival

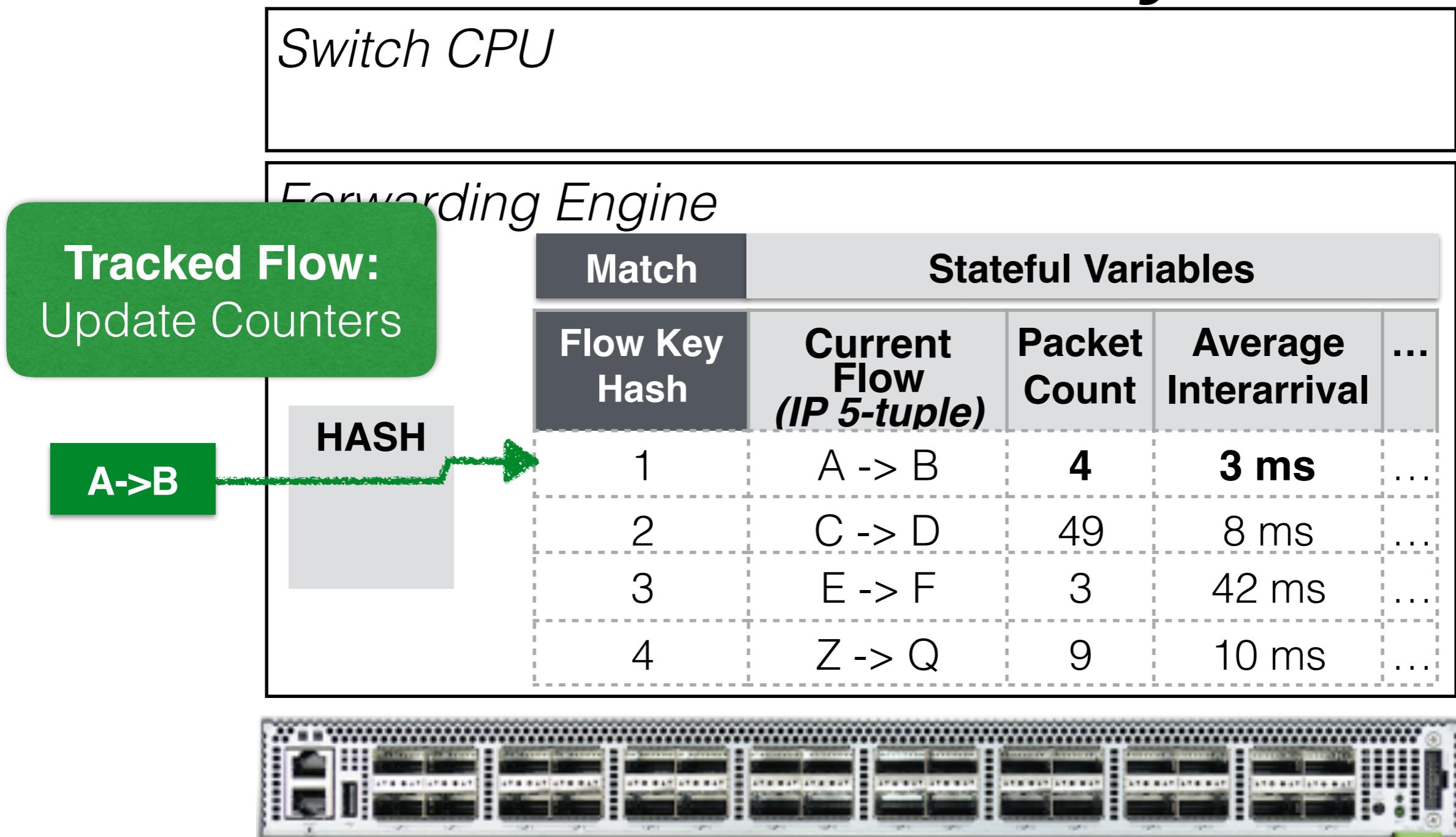
...



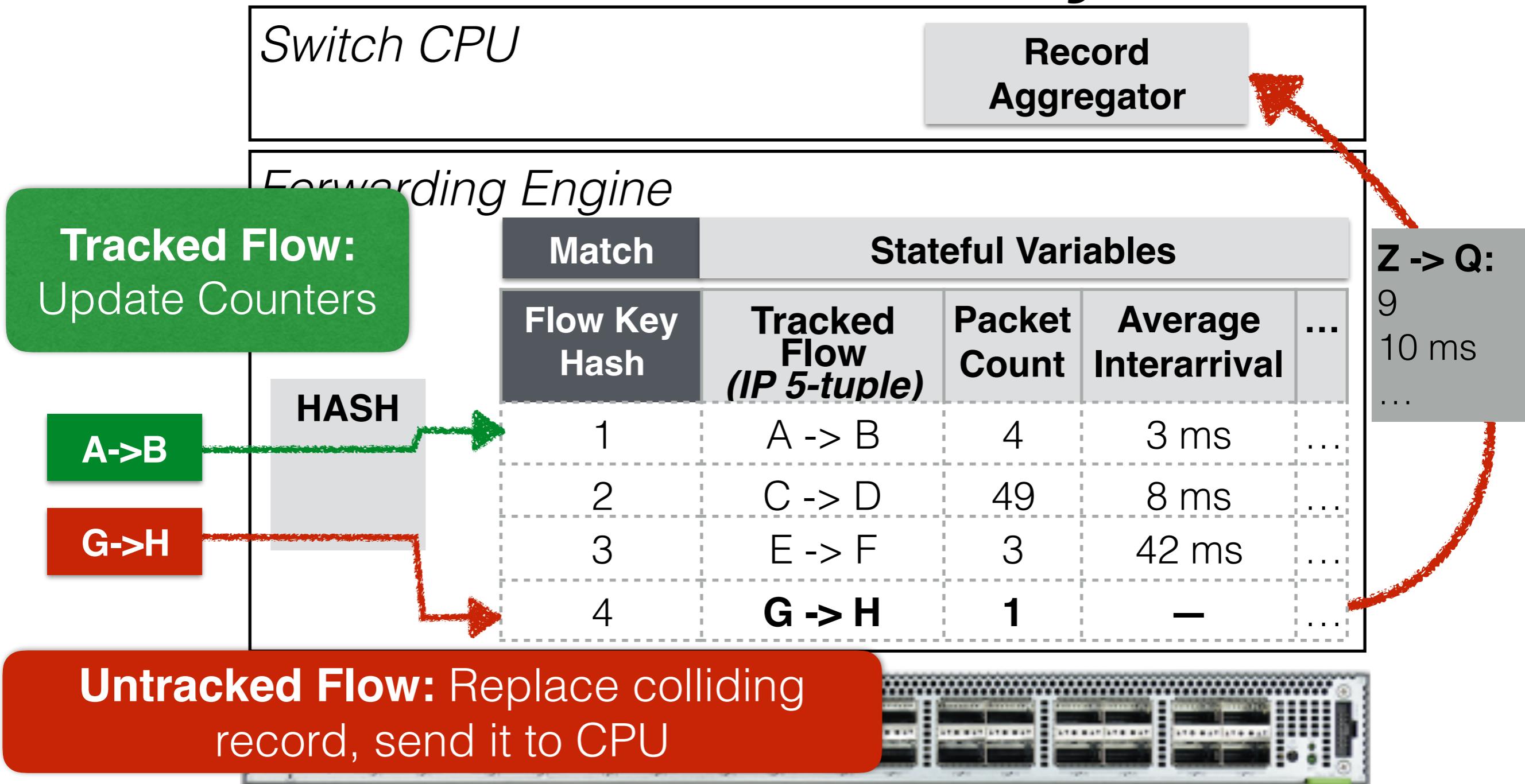
TurboFlow Architecture: Using the FE Efficiently



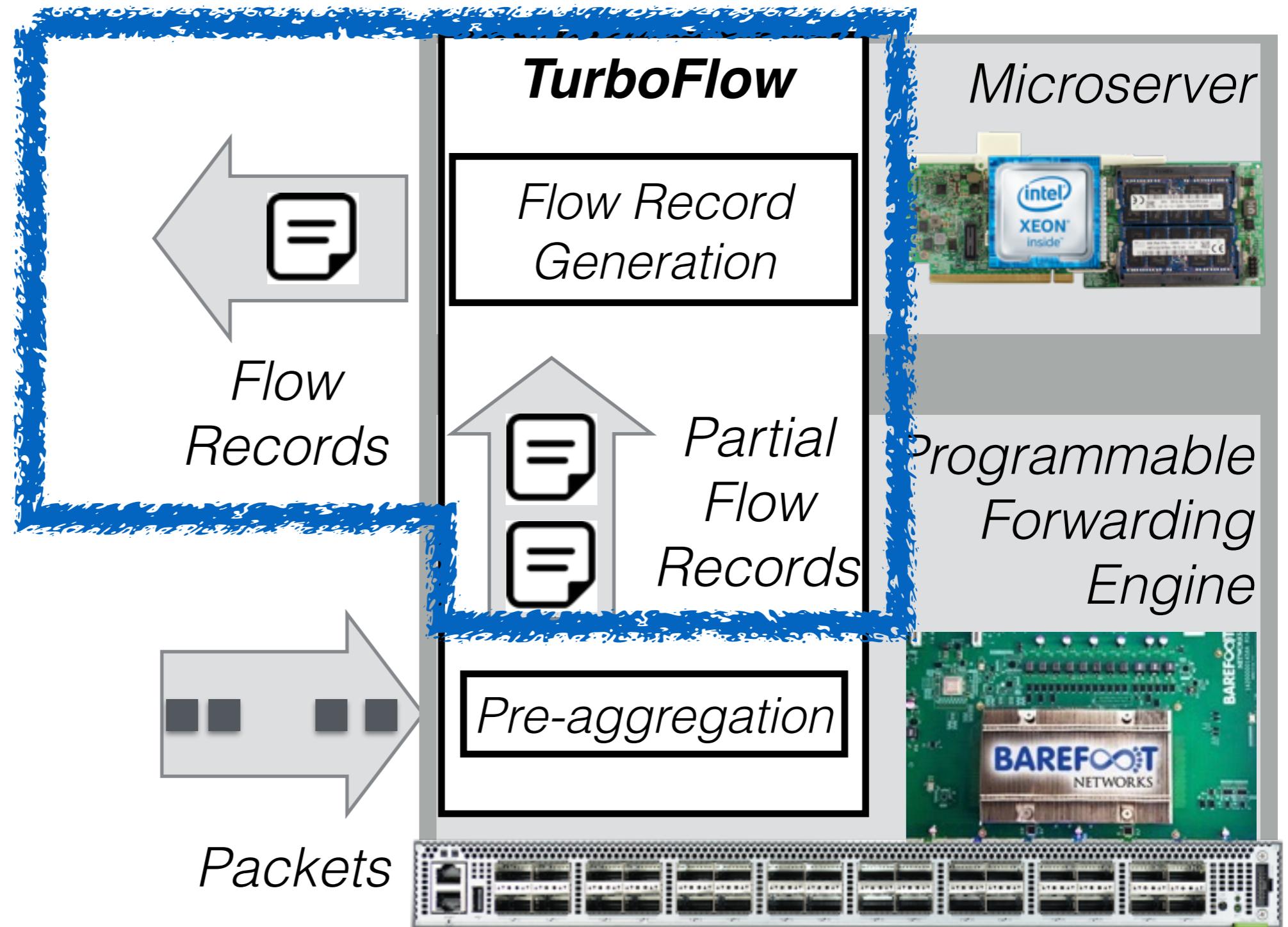
TurboFlow Architecture: Using the FE Efficiently



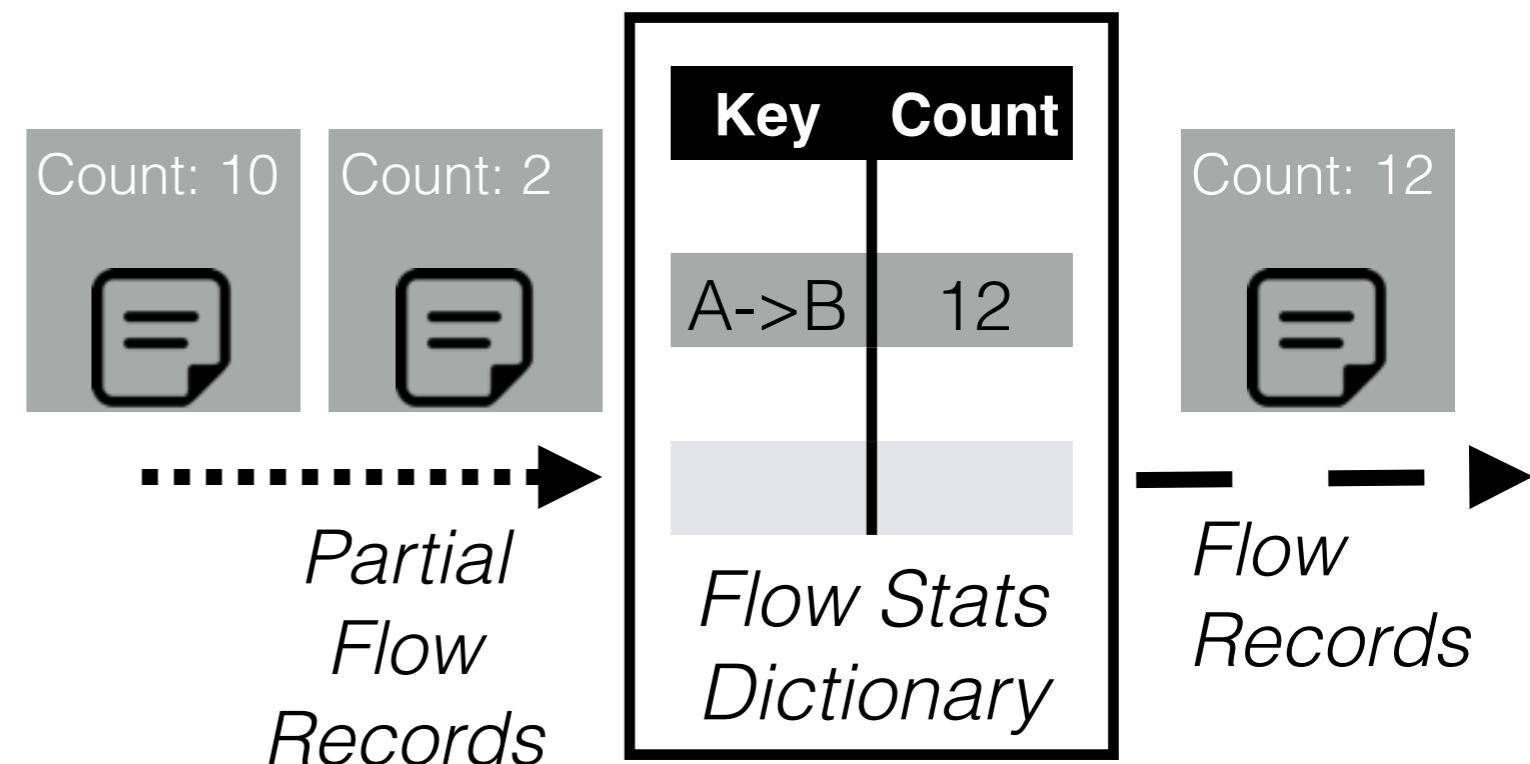
TurboFlow Architecture: Using the FE Efficiently



TurboFlow Design

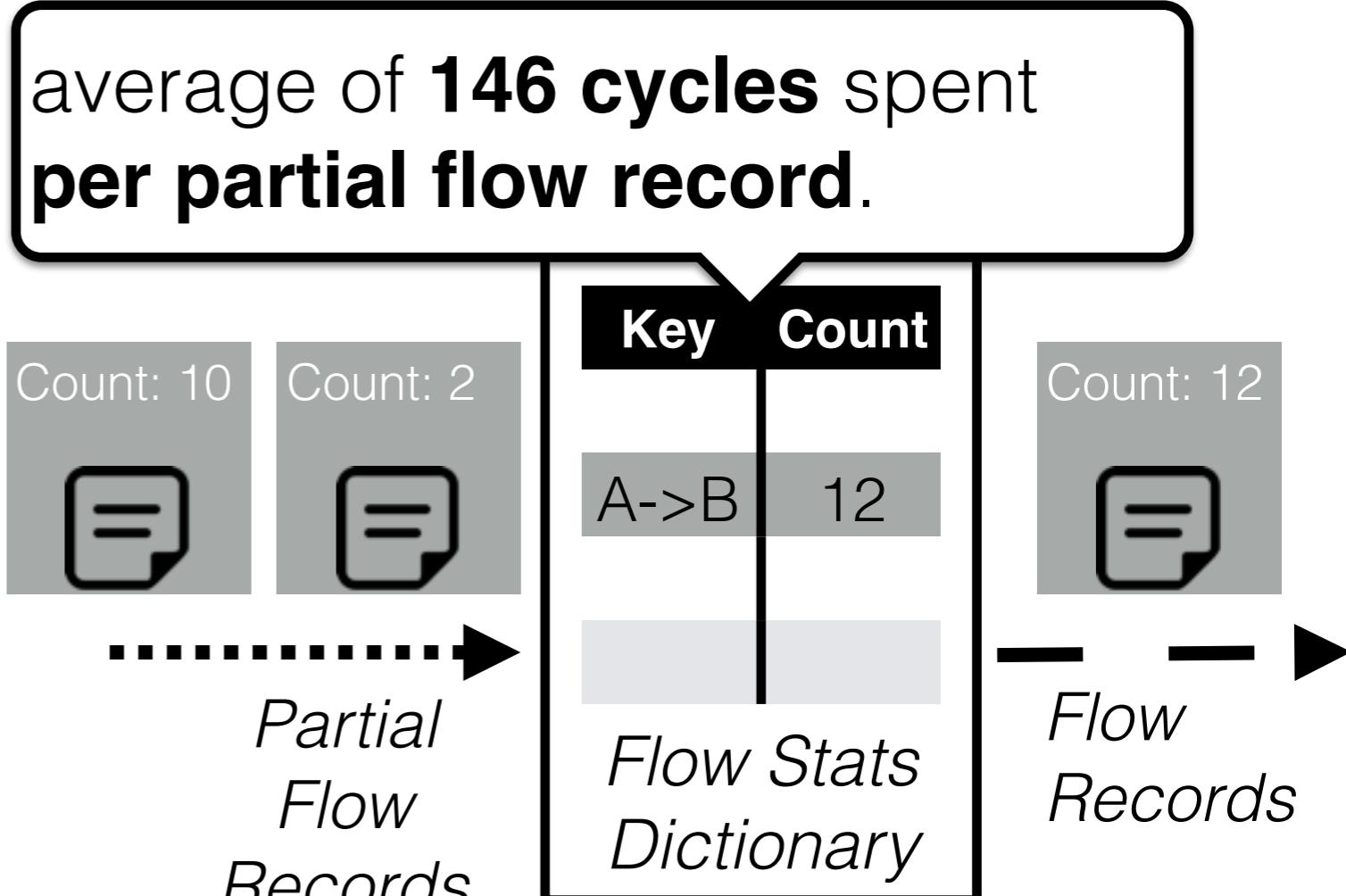


TurboFlow Architecture: Using the CPU Efficiently



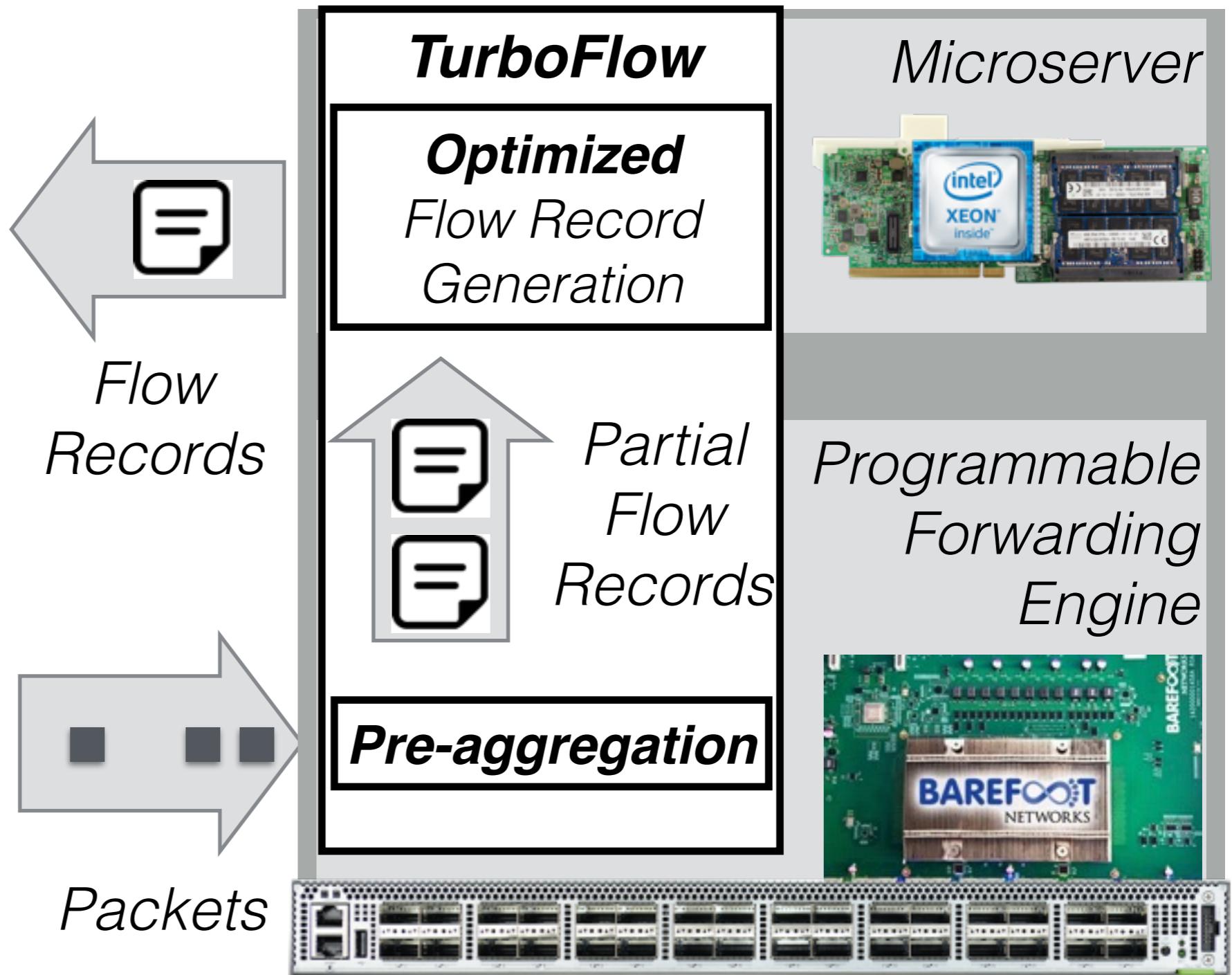
TurboFlow Architecture: Using the CPU Efficiently

Optimization	Performance Vs. Baseline
baseline (<code>std::unordered_map</code>)	-
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

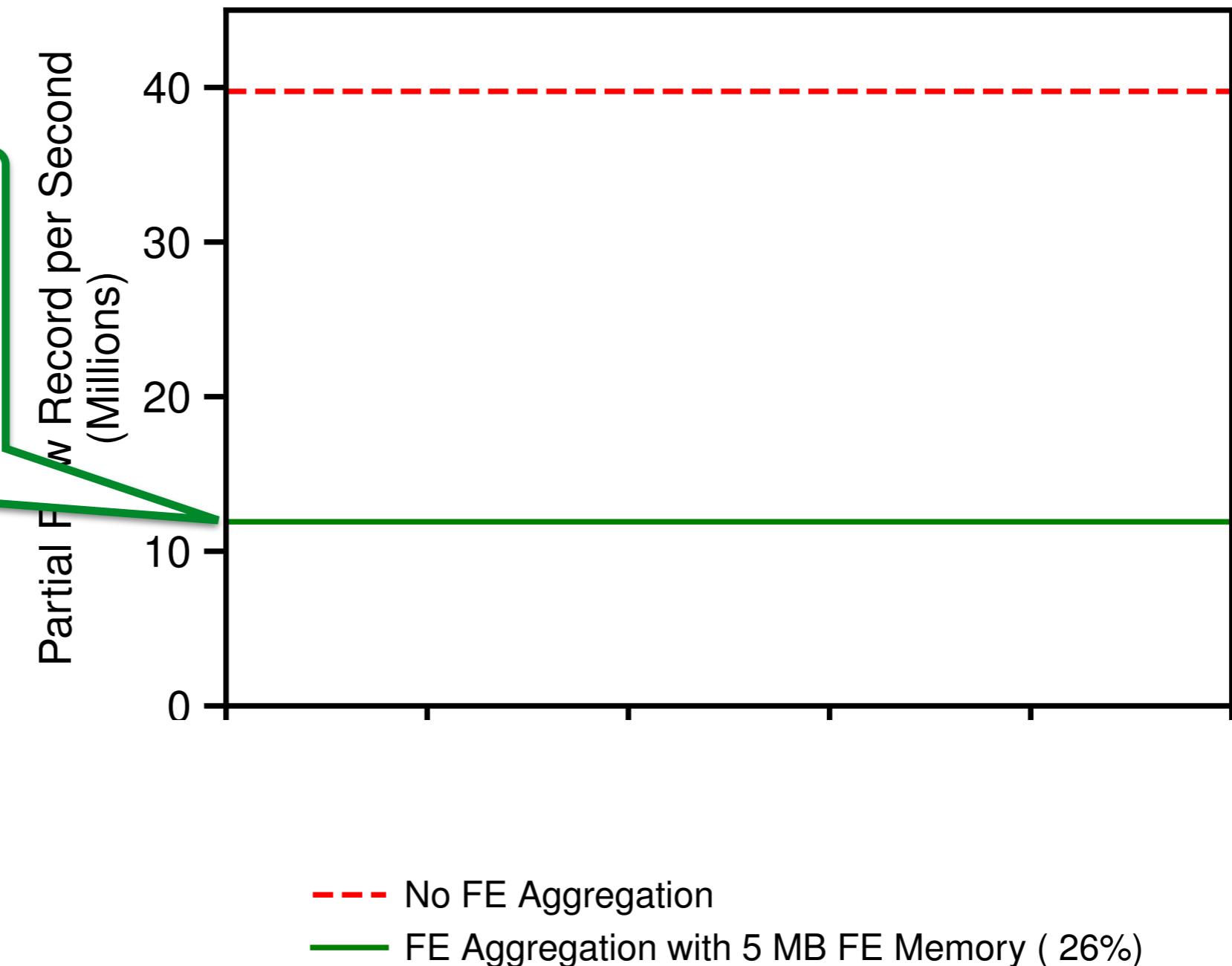
Implementations	Benchmark Workloads
<p data-bbox="322 733 1133 808">P4 Switch</p> <p data-bbox="166 870 1325 972">(3.2 Tb/s Barefoot Tofino)</p> 	<ul data-bbox="1434 733 2636 1157" style="list-style-type: none"><li data-bbox="1434 733 2636 1157">• 10 Gb/s Internet Router Traces (CAIDA 2015)
<p data-bbox="423 1218 1051 1300">P4 SmartNIC</p> <p data-bbox="131 1361 1352 1464">(40 Gb/s Netronome NFP)</p> 	<ul data-bbox="1434 1341 2636 1751" style="list-style-type: none"><li data-bbox="1434 1341 2636 1751">• 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)

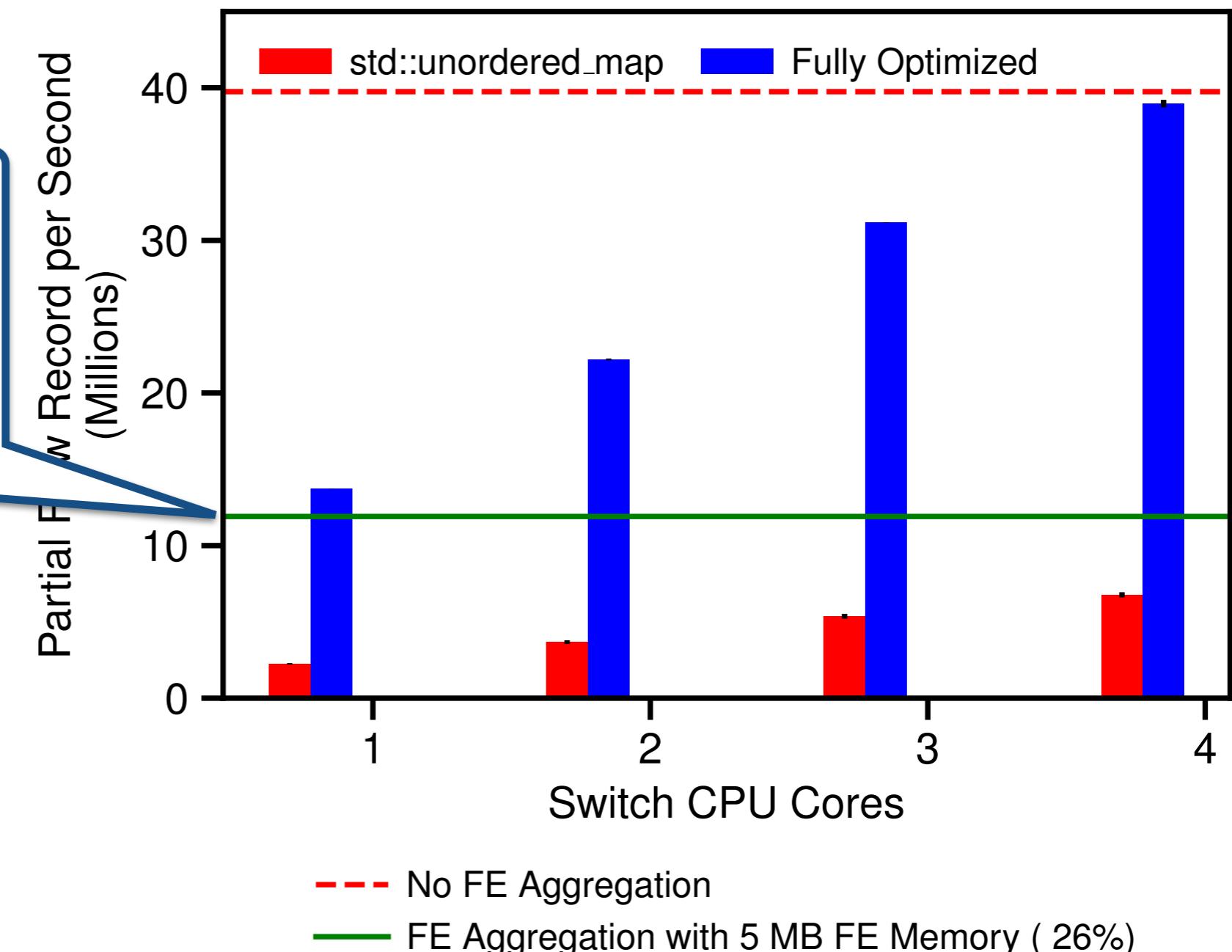
Required Average Throughput to Monitor 100 X 10 Gb/s Internet Links

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



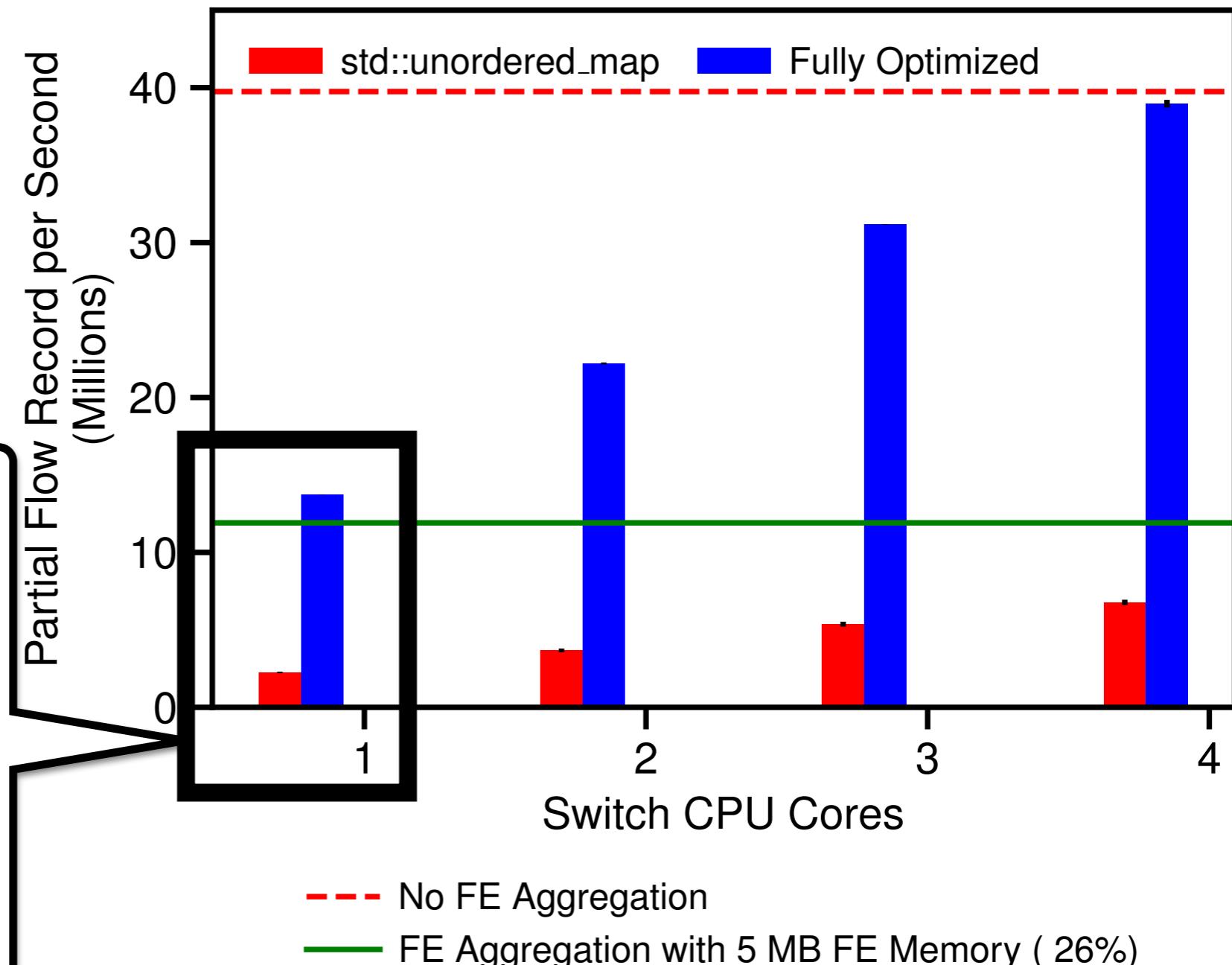
Required Average Throughput to Monitor 100 X 10 Gb/s Internet Links

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

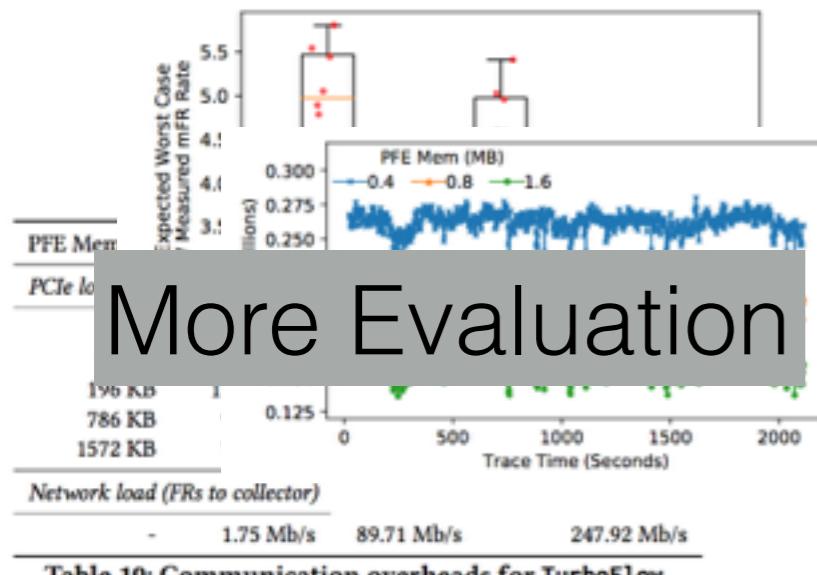


Table 10: Communication overheads for TurboFlow.

Feature Type	Examples	Applications
<i>Traffic Characteristics</i>		
Metadata	QoS type, IP options, TCP options & flags	Security [84], flow scheduling [2, 41], auditing [50], heavy hitter detection [91], QoS monitoring [62]
Statistics	duration, packet count, byte count, jitter, max packet size	

More interesting flow features

Table 2: Types of FR features and example applications.

Workload	Switches	+ Generation	+ Analysis
<i>Equipment Cost (per Tb/s)</i>			
DC ToR	\$3600	\$3603 (+ 0.1%)	\$3642 (+ 1.2%)
DC Agg.	\$3600	\$3608 (+ 0.2%)	\$3702 (+ 2.9%)
Internet			9 (+ 12.8%)
<i>Power Cost</i>			
DC ToR	150 W	158 W (+ 5.6%)	164 W (+ 10.0%)
DC Agg.	150 W	159 W (+ 6.1%)	174 W (+ 16.7%)
Internet	150 W	163 W (+ 9.2%)	234 W (+ 56.3%)

Table 12: Cost of monitoring infrastructure with TurboFlow.

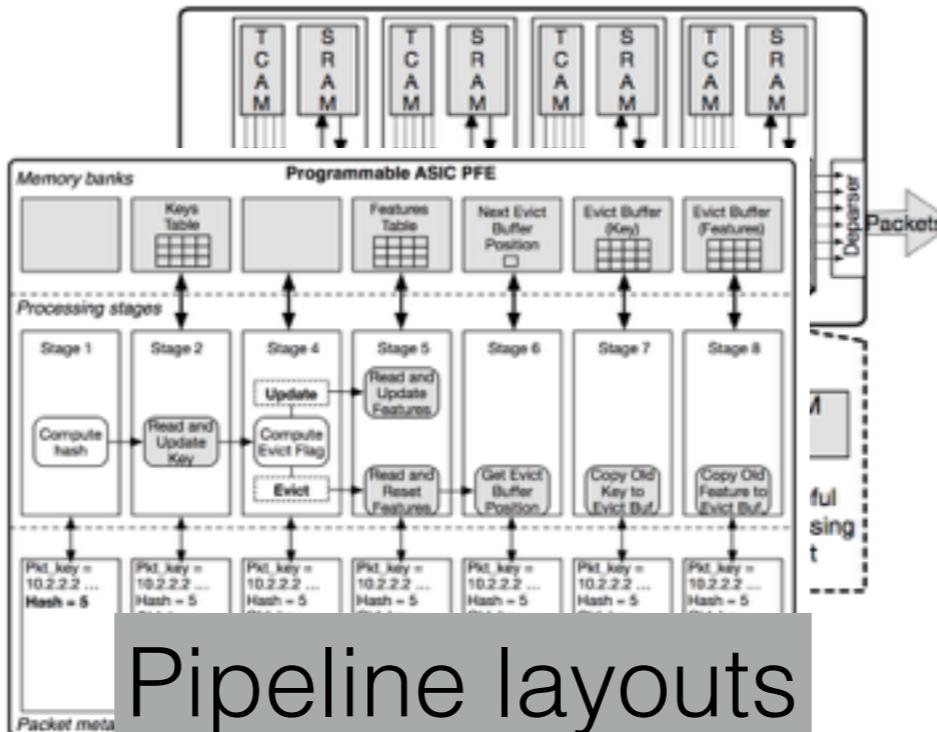


Figure 6: MFR generator mapped to a programmable ASIC.

```
// Tables.
table UpdateKey { default_action :UpdateKeyAction(); }
table UpdateFeatures { default_action :UpdateFeaturesAction(); }
table ResetFeatures { default_action :ResetFeaturesAction(); }
```

```
// Actions.
// Update key for every packet.
action UpdateKeyAction() {
    (pkt.key string* tempMfr.key));
}

// Update features when there is no collision.
action UpdateFeaturesAction() {
    register_read(tempMfr.pktCt, pktCtArr, md.hash);
    register_write(pktCtArr, md.hash, tempMfr.pktCt+1);
}

// Reset features and evict on collision.
action ResetFeaturesAction() {
    register_read(tempMfr.pktCt, pktCtArr, md.hash);
    register_write(pktCtArr, md.hash, 1);
    register_read(tempMfr.evictBufPos, evictBufArr, 0);
    register_write(evictBufArr, 0, tempMfr.evictBufPos+1);
    register_write(evictBufKey, tempMfr.evictBufPos,
        tempMfr.key);
    register_write(evictBufPktCt, tempMfr.evictBufPos,
        tempMfr.pktCt);
}
```

$$P[\text{eviction}] = 1 - (1 - \frac{1}{T})^{\hat{A}}$$

$$E[m] = E[f] + (E[p] - E[f]) * P[\text{eviction}]$$

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**.

