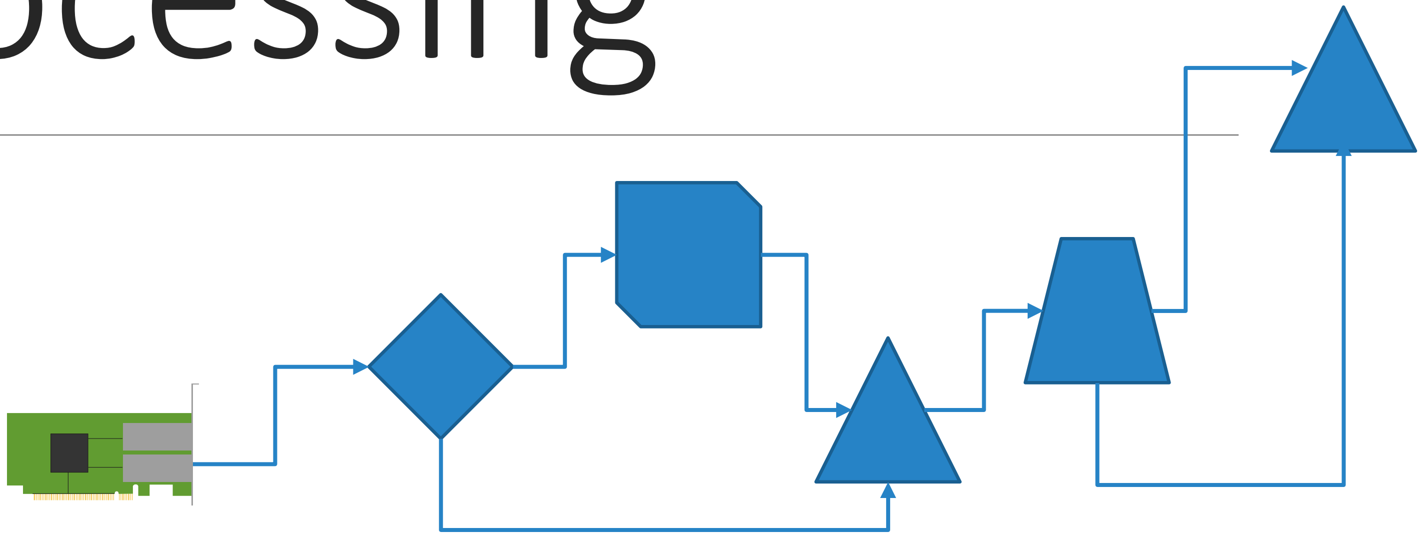
High-speed stateful packet processing

PR. TOM BARBETTE



Network Functions Are Pervasive

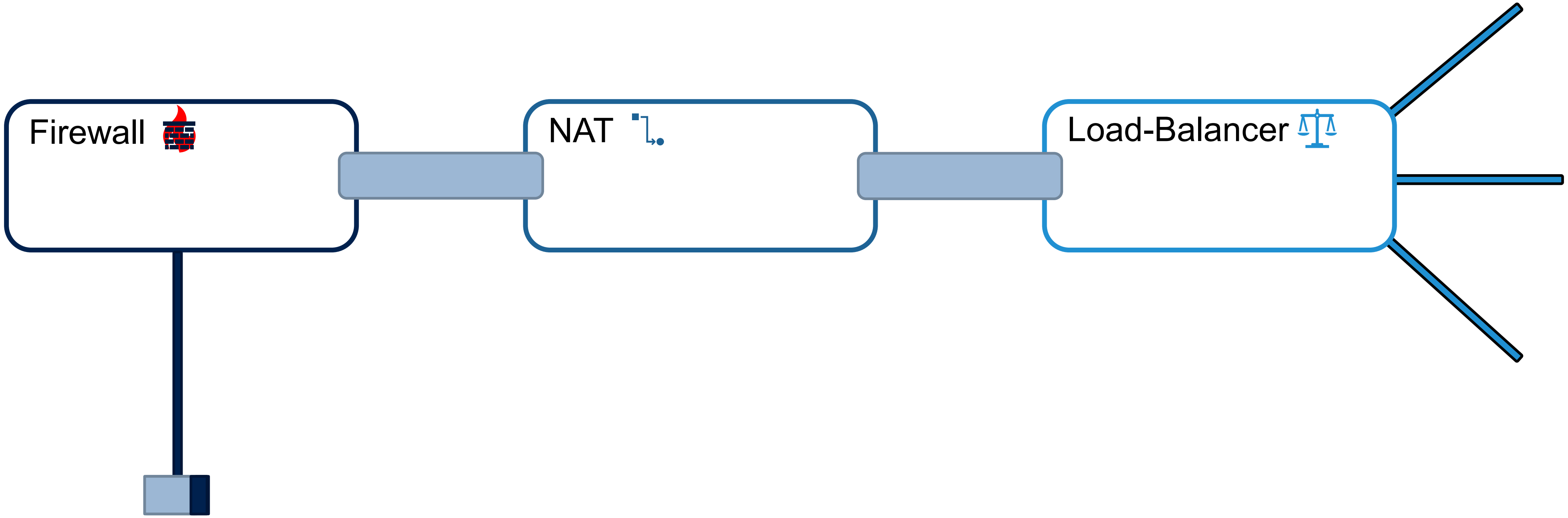
Firewall

NAT

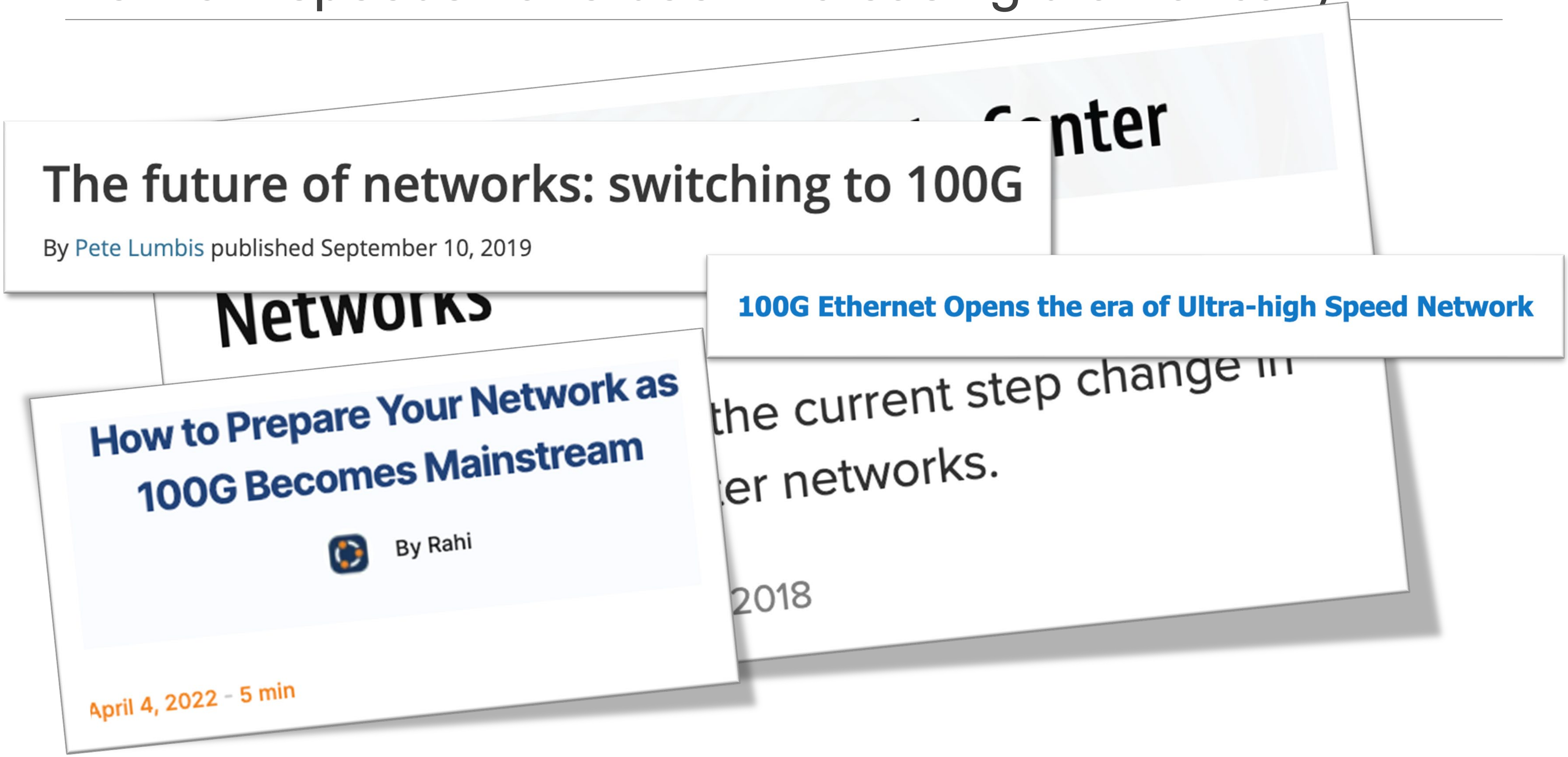
Load-Balancer

Network Functions Are Pervasive

Network Functions Virtualization is an essential architectural paradigm of today's networks



Network speeds have been increasing dramatically



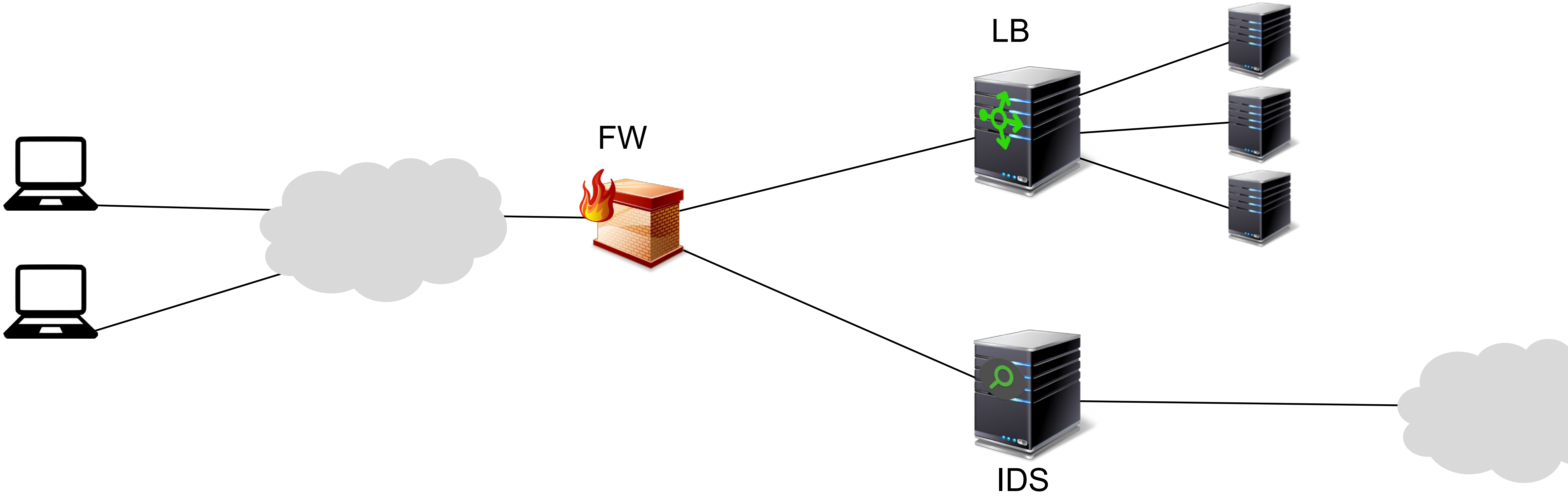
Talk goal

How to handle stateful network functions for terabit per second of traffic?

- High-speed stateful software packet processing
- Switch-assisted stateful packet processing
- NIC-assisted stateful packet processing
- NFV service chain combined stateful packet processing

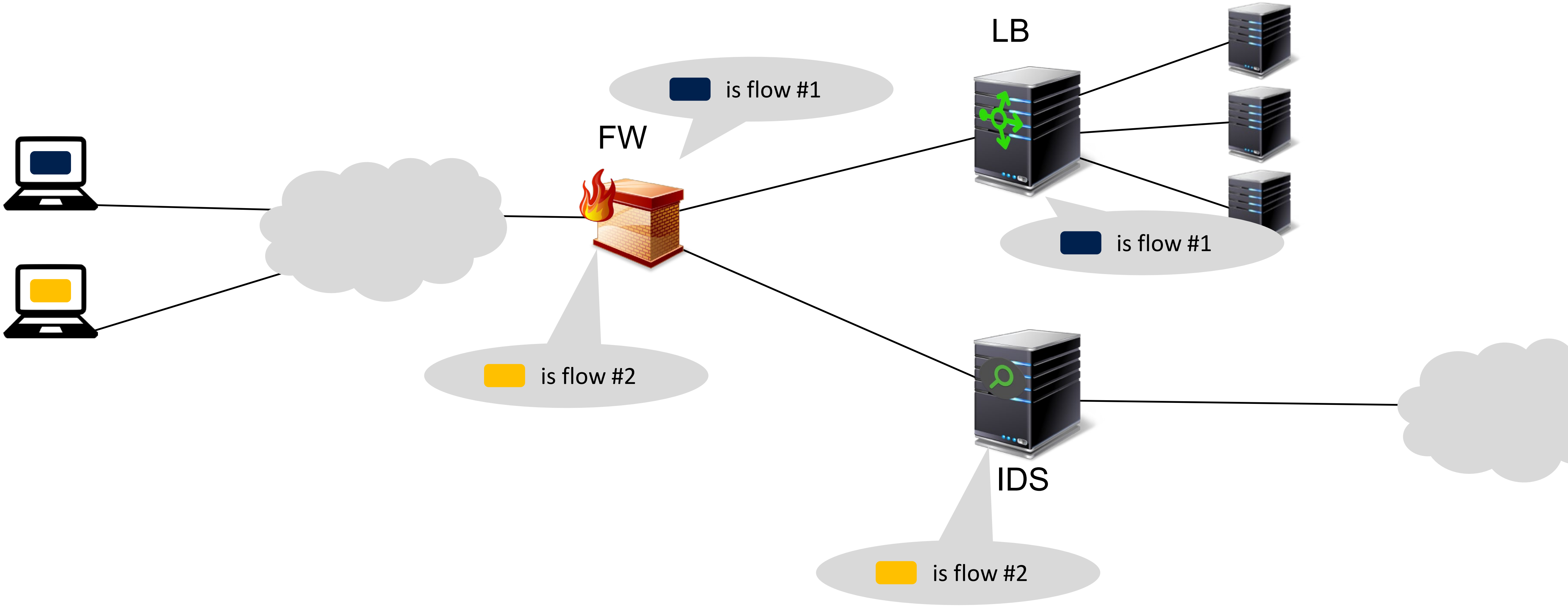
Connection Tracking

Connection tracking is about classifying packets in micro-flows



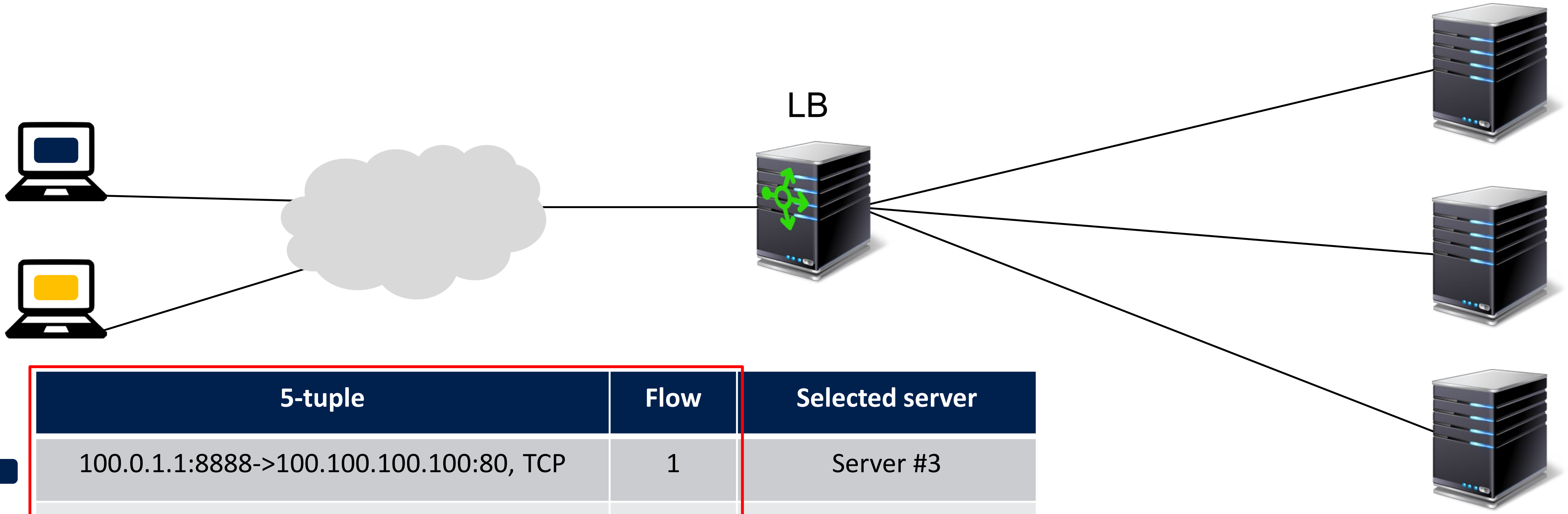
Connection Tracking

Connection tracking is about classifying packets in micro-flows



Connection Tracking in a (stateful) Load Balancer

Send all the flow's packets to the same server



5-tuple	Flow	Selected server
100.0.1.1:8888->100.100.100.100:80, TCP	1	Server #3
100.0.1.2:9999->100.100.100.100:80, TCP	2	Server #1

Connection Tracking

Challenges in High-speed Connection Tracking

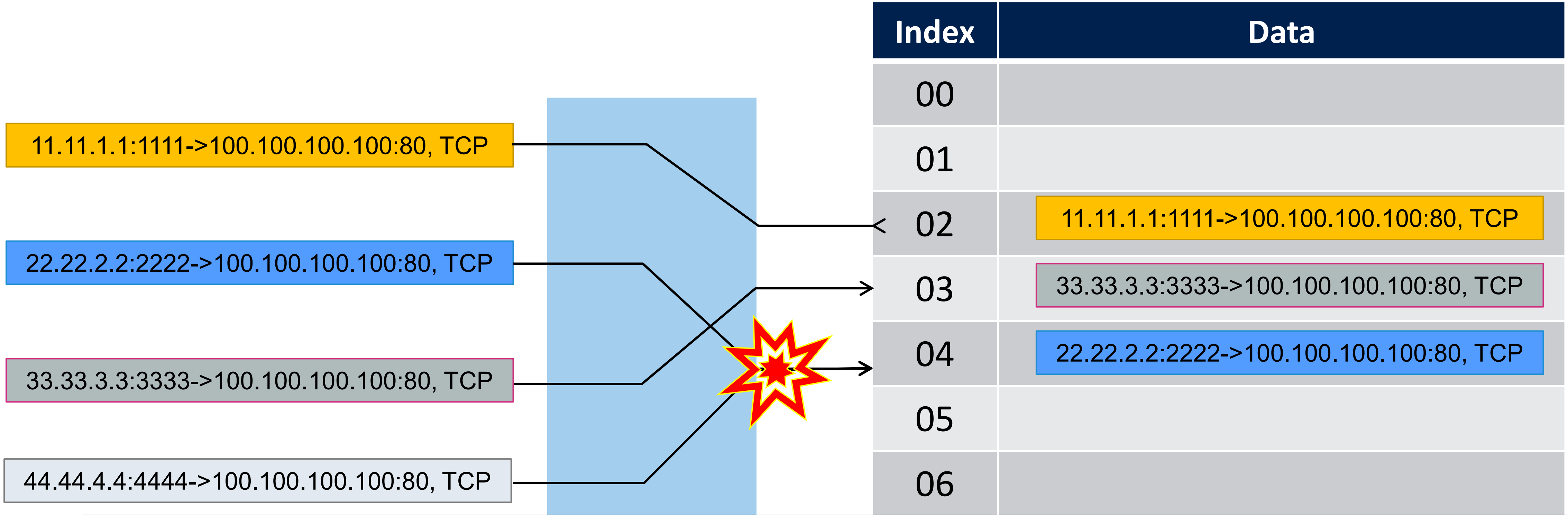
- On a 100 Gbps link, packets arrive every 6.72 ns
- A DRAM access takes ~100 ns

Data structures must leverage CPU caches

How do we build 100GbE+ software stateful Network Functions?

Hash Tables (HT) in a nutshell

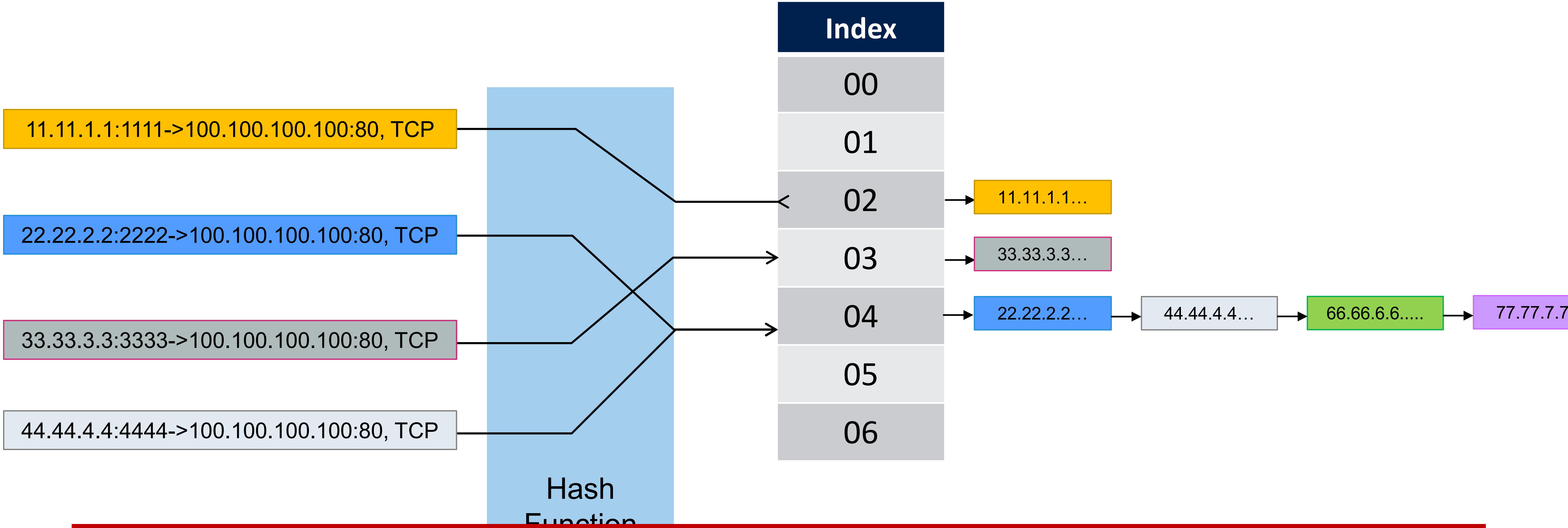
Hash Tables (HT) in a nutshell



HT implementations differ for collision handling

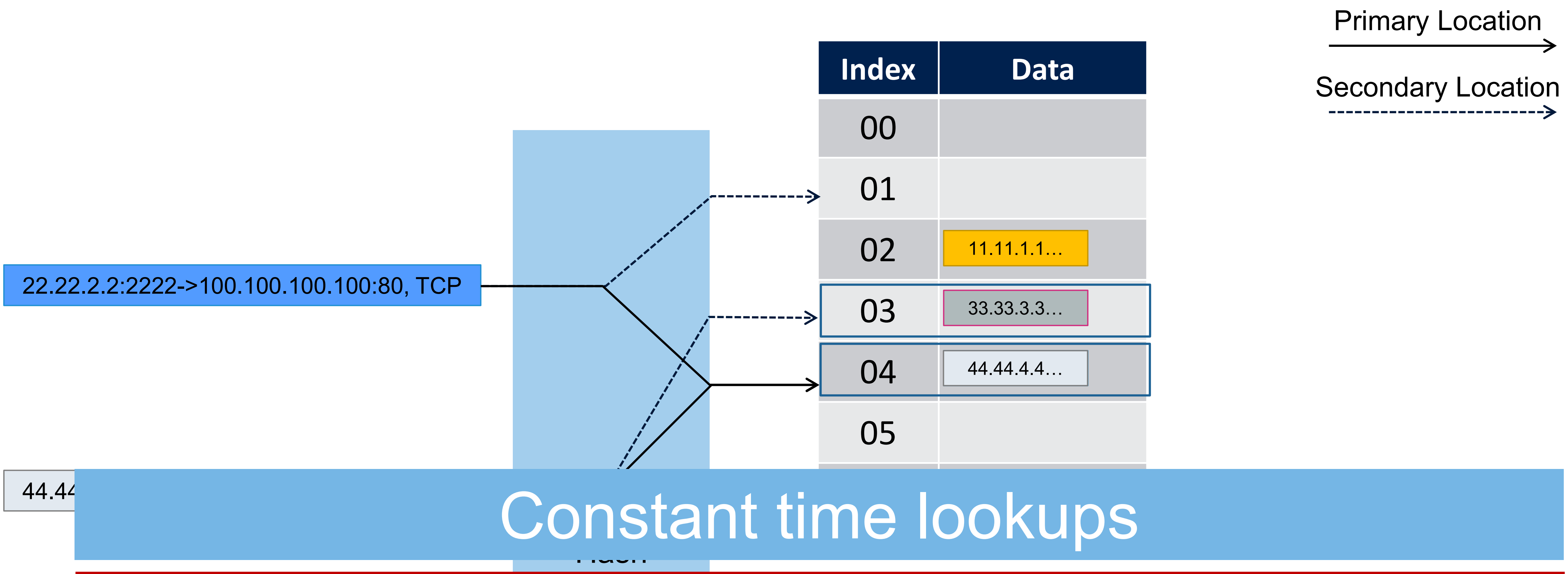
Perfect Hash Functions are difficult to implement

Chaining Hash Tables



Lists can grow infinitely!

Cuckoo Hash Tables

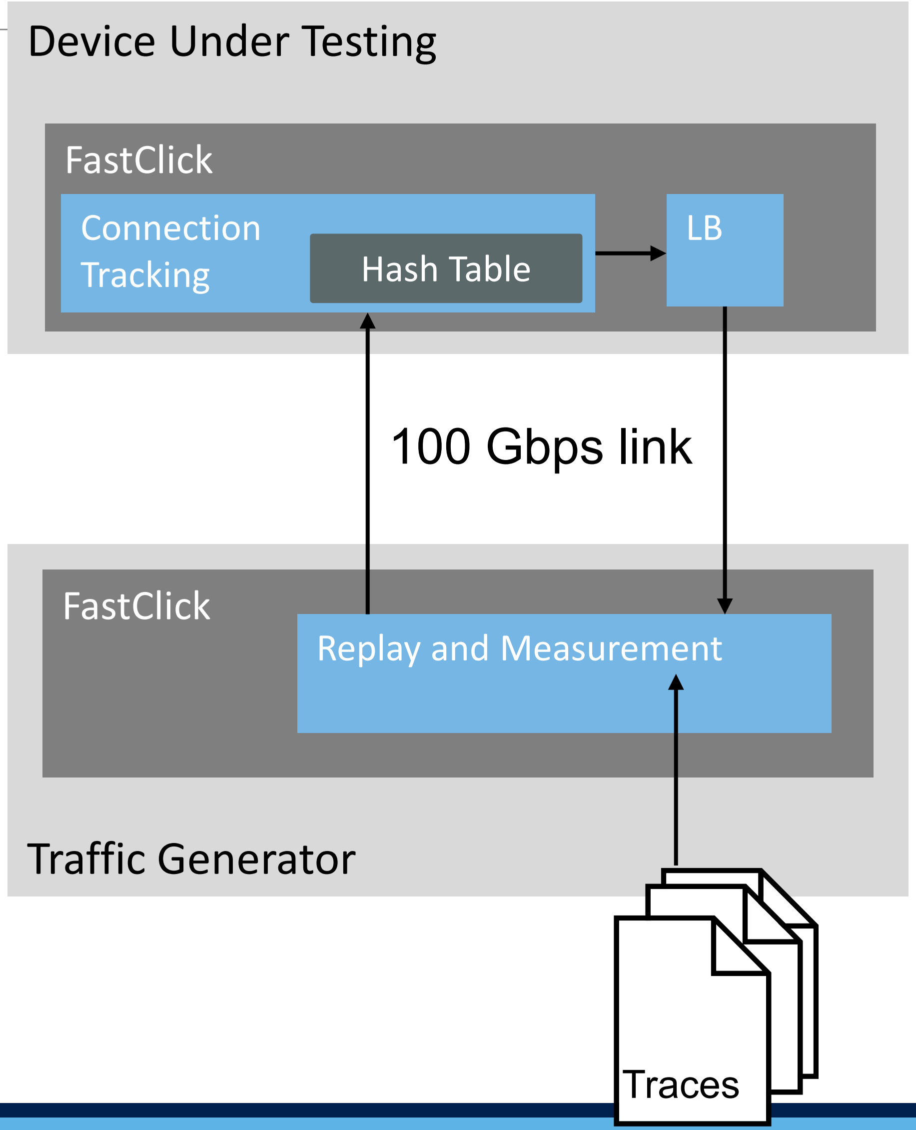


Constant time lookups

Under high load, long swapping chain

Testbed

- 2 server machines connected back-to-back
- Mellanox ConnectX-5 @ 100Gbps
- FastClick Stateful Load Balancer configuration
- Traces captured at our campus and CAIDA
- Multiple parallel replays

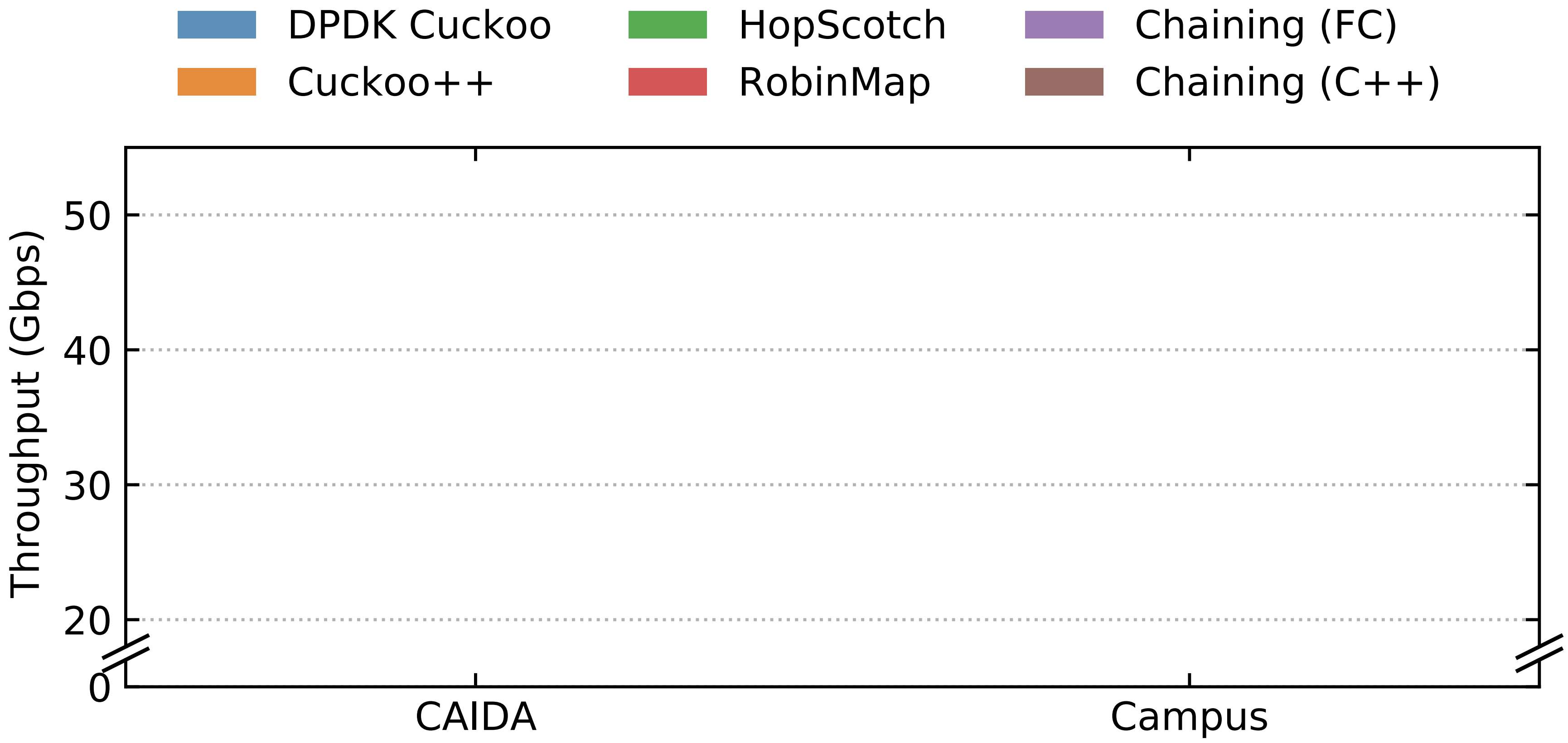


[HPSR’21] Girondi, M Chiesa, T Barbette

Studied Hash Tables

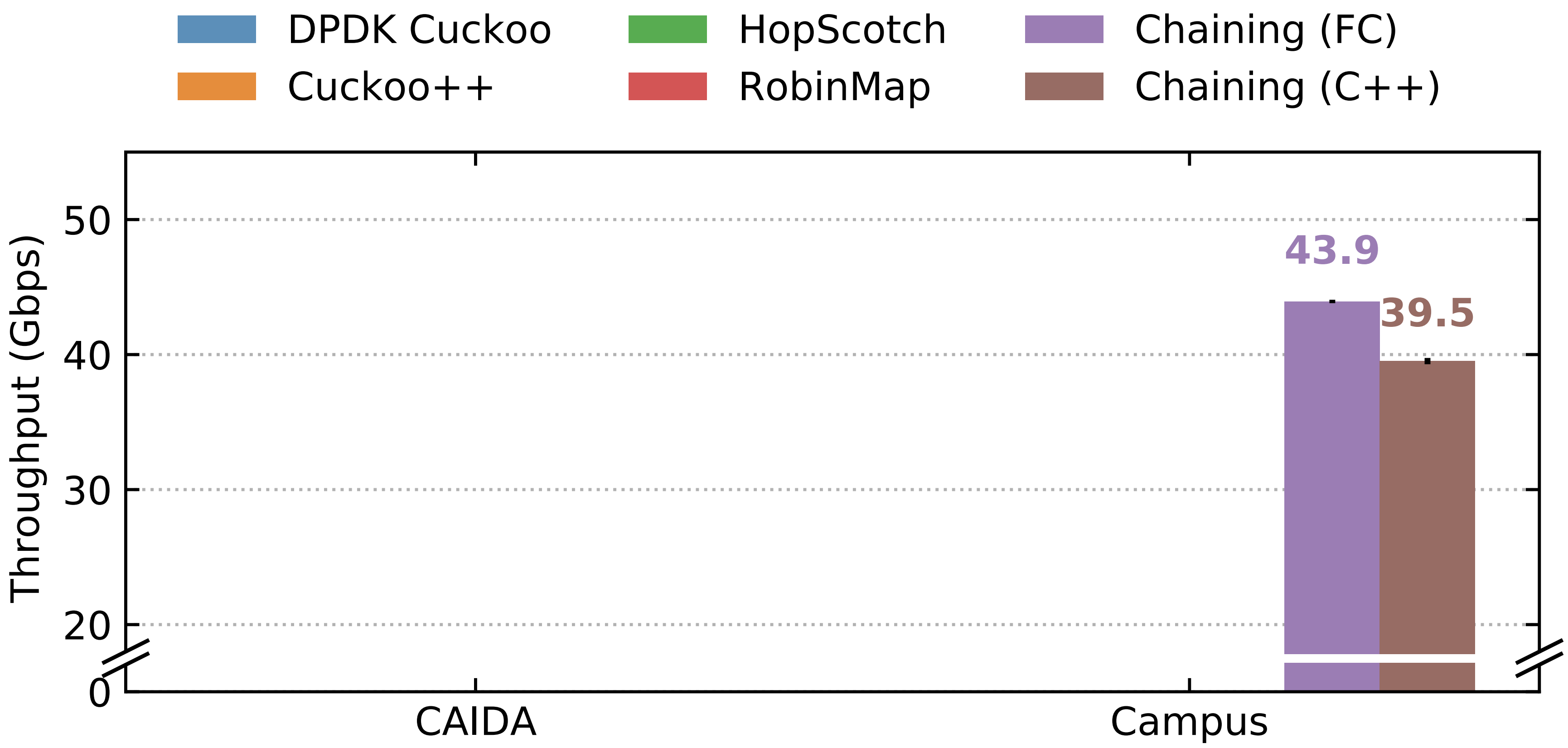
N. Le Scouarnec, **Cuckoo++ Hash Tables** in ANCS 2018
M. Herlihy et al., **Hopscotch hashing** in DISC 2008
P. Celis et al., **Robin hood hashing** in SFCS 1985

Single core: throughput



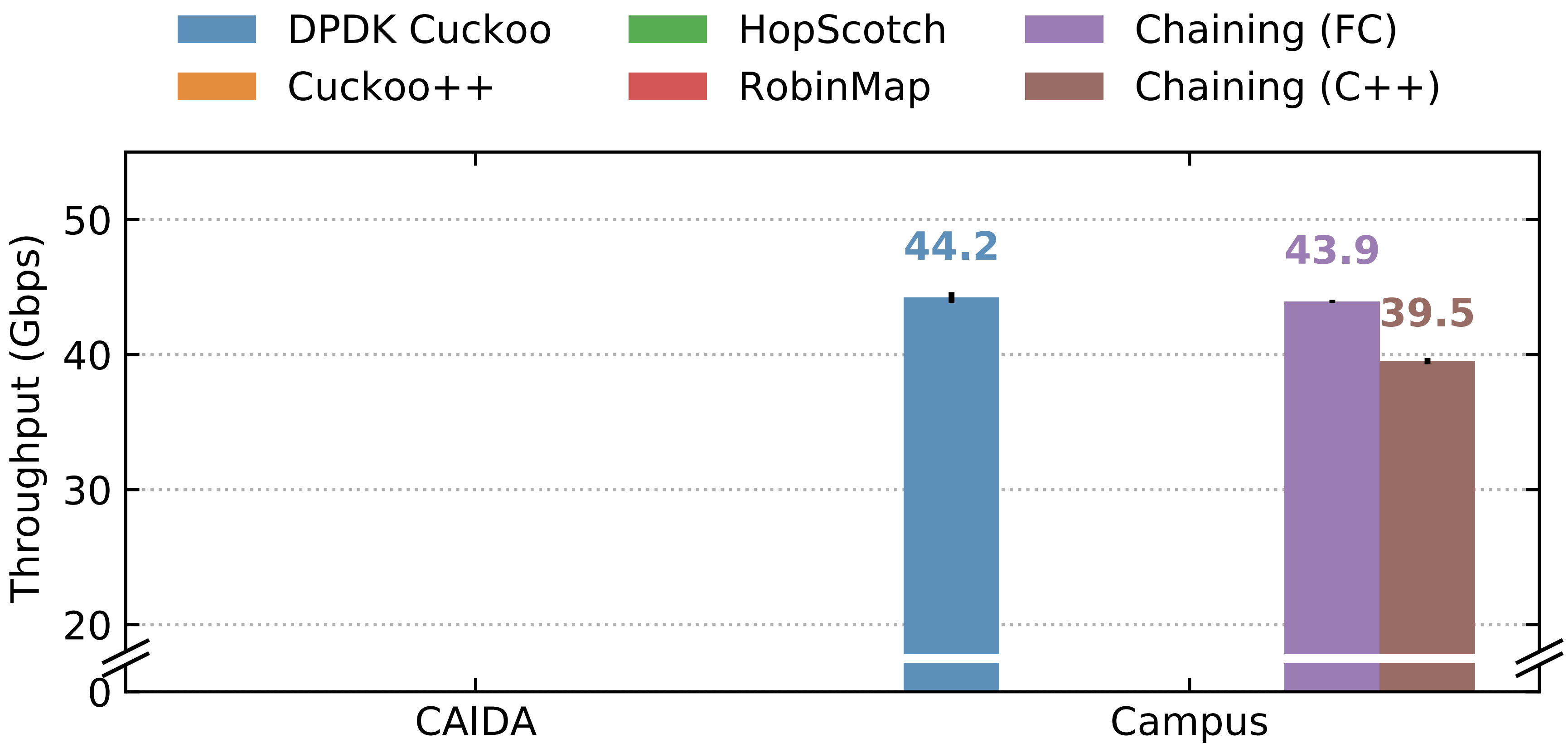
4M entries hash table, ≈ 55 Gbps

Single core: throughput



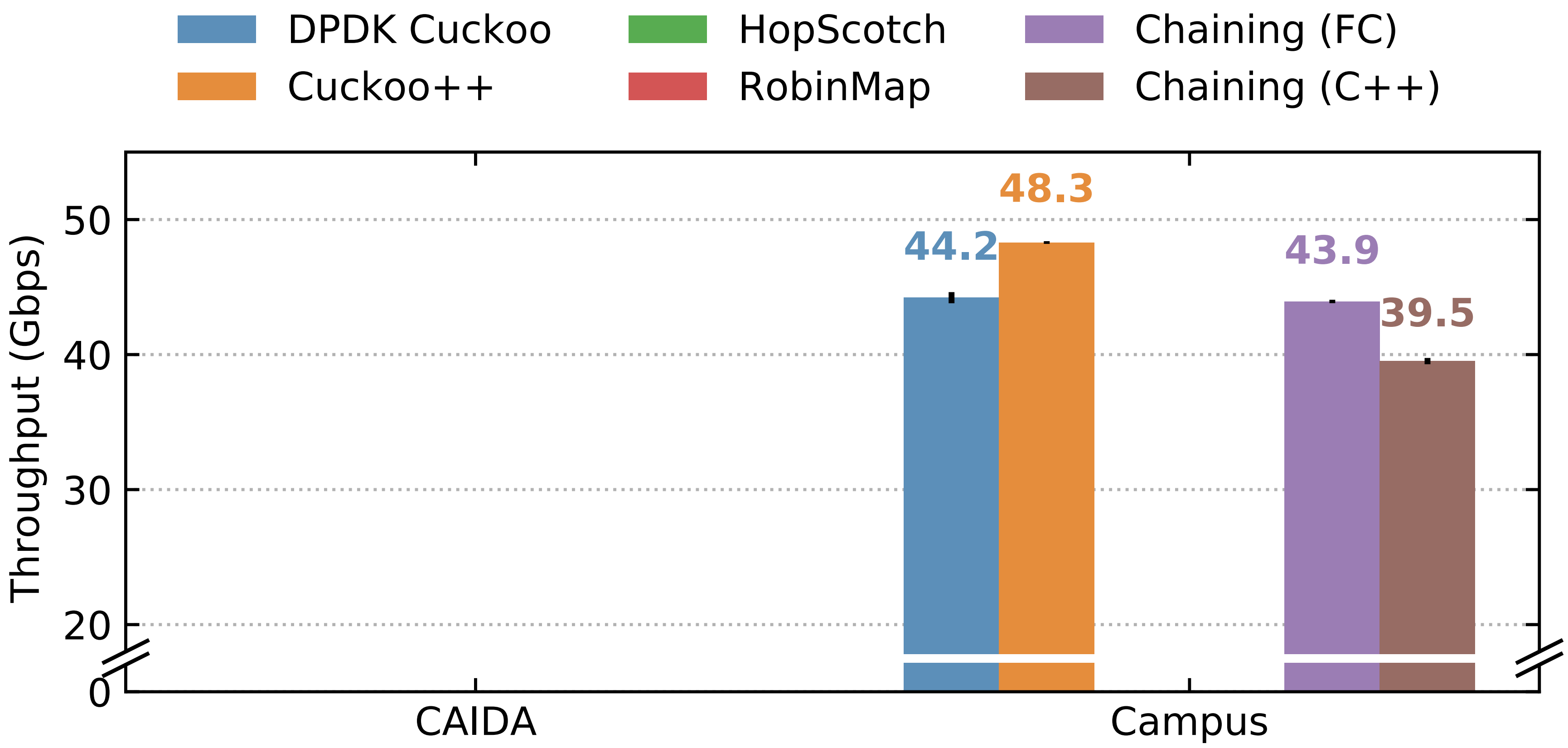
4M entries hash table, ≈ 55 Gbps

Single core: throughput



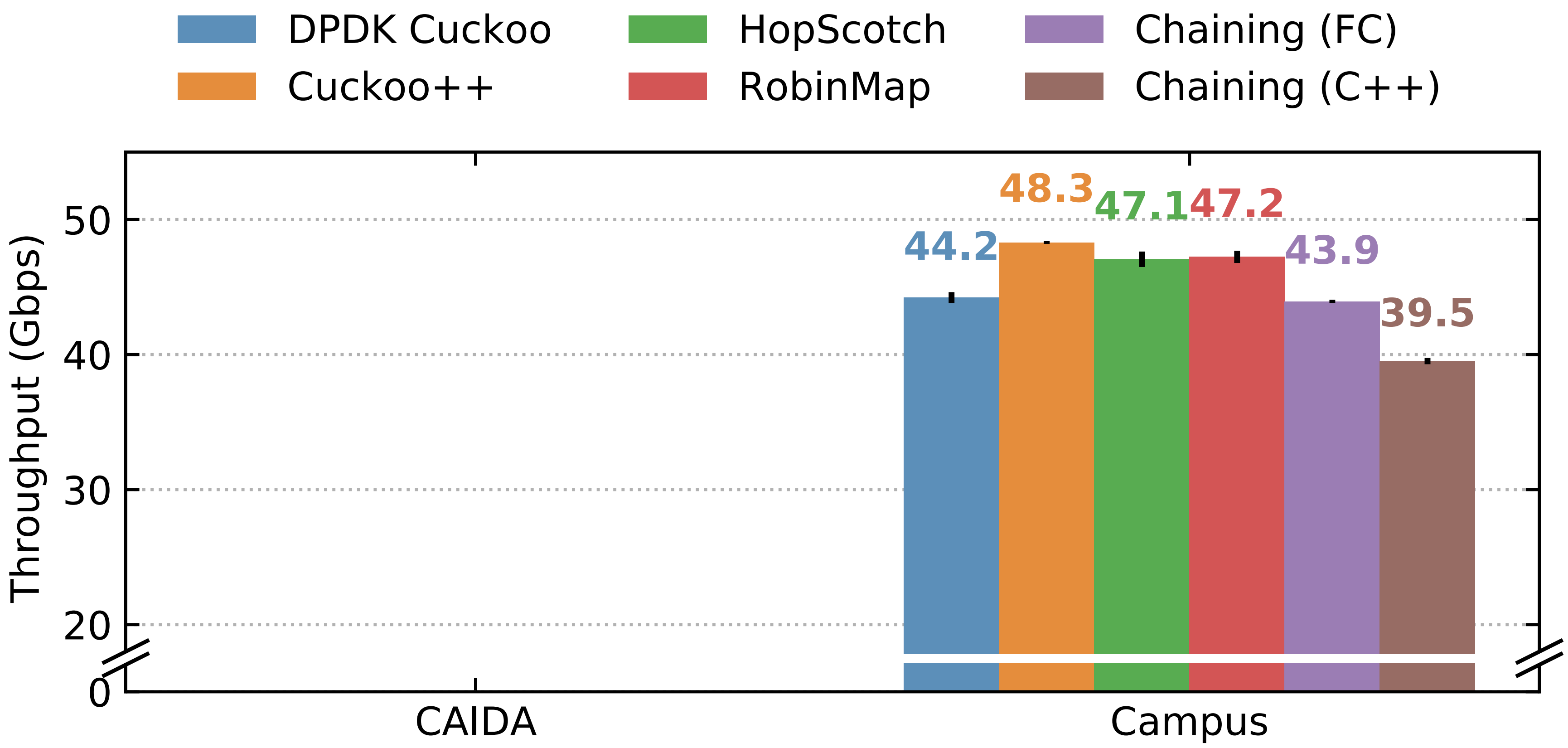
4M entries hash table, ≈ 55 Gbps

Single core: throughput



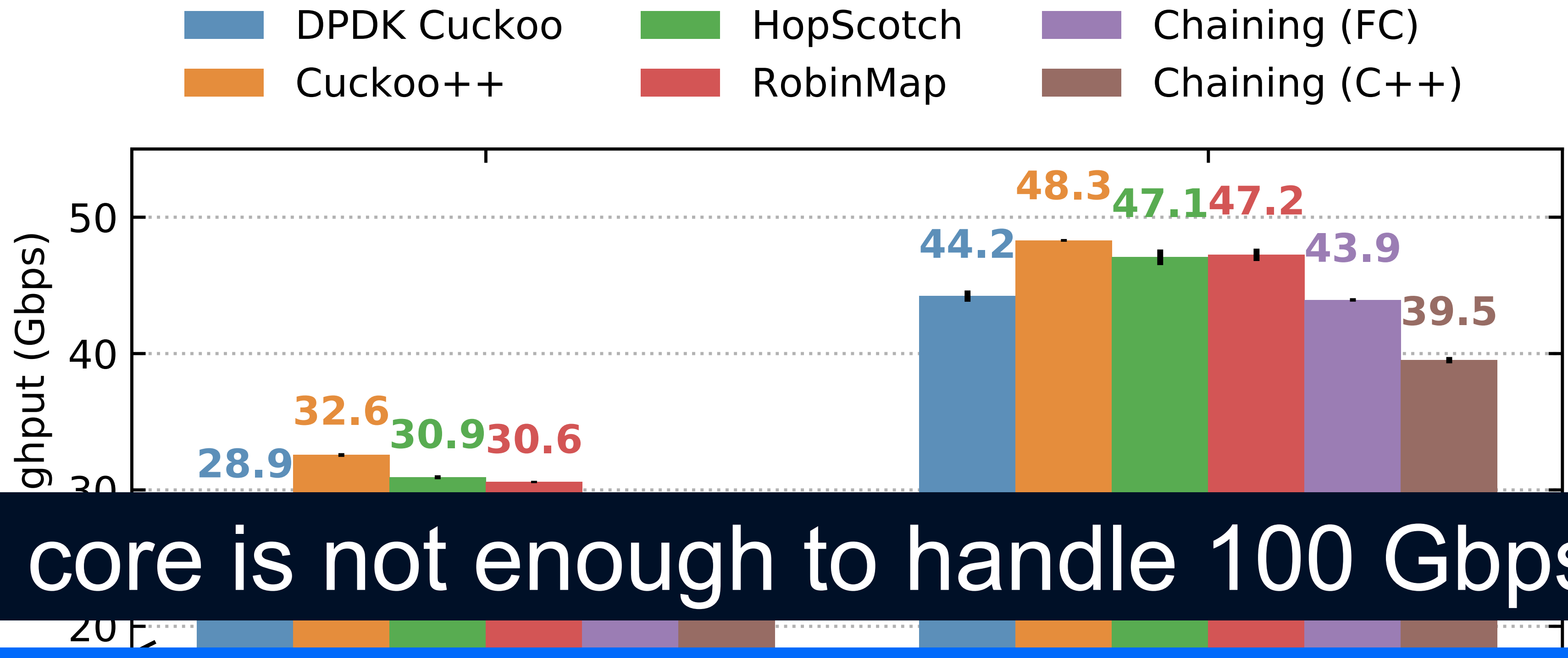
4M entries hash table, ≈ 55 Gbps

Single core: throughput



4M entries hash table, ≈ 55 Gbps

Single core: throughput



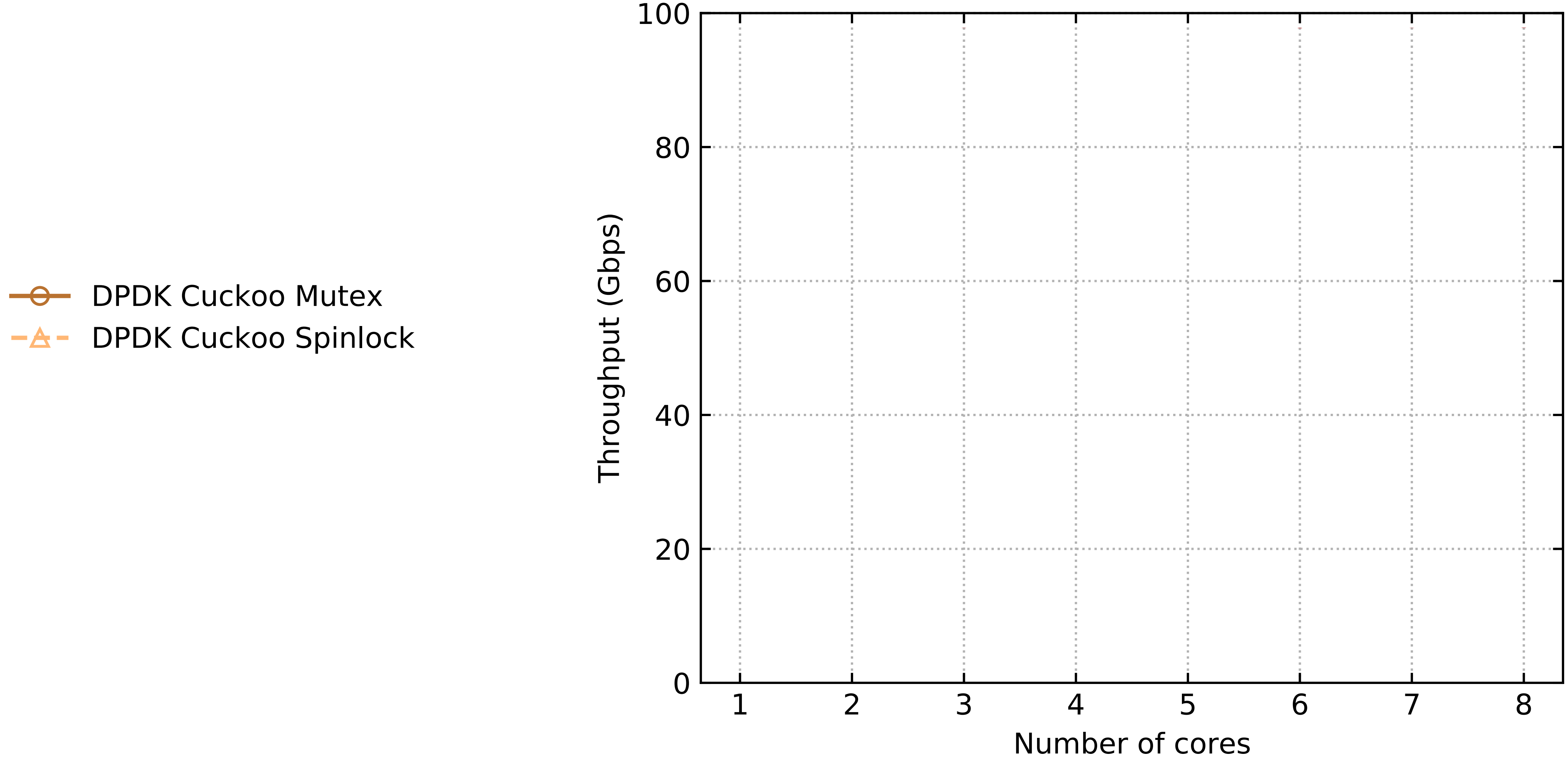
1 core is not enough to handle 100 Gbps

Study of Insertion and Lookup cycles in the paper

CAIDA has ≈ twice flows, similar trends

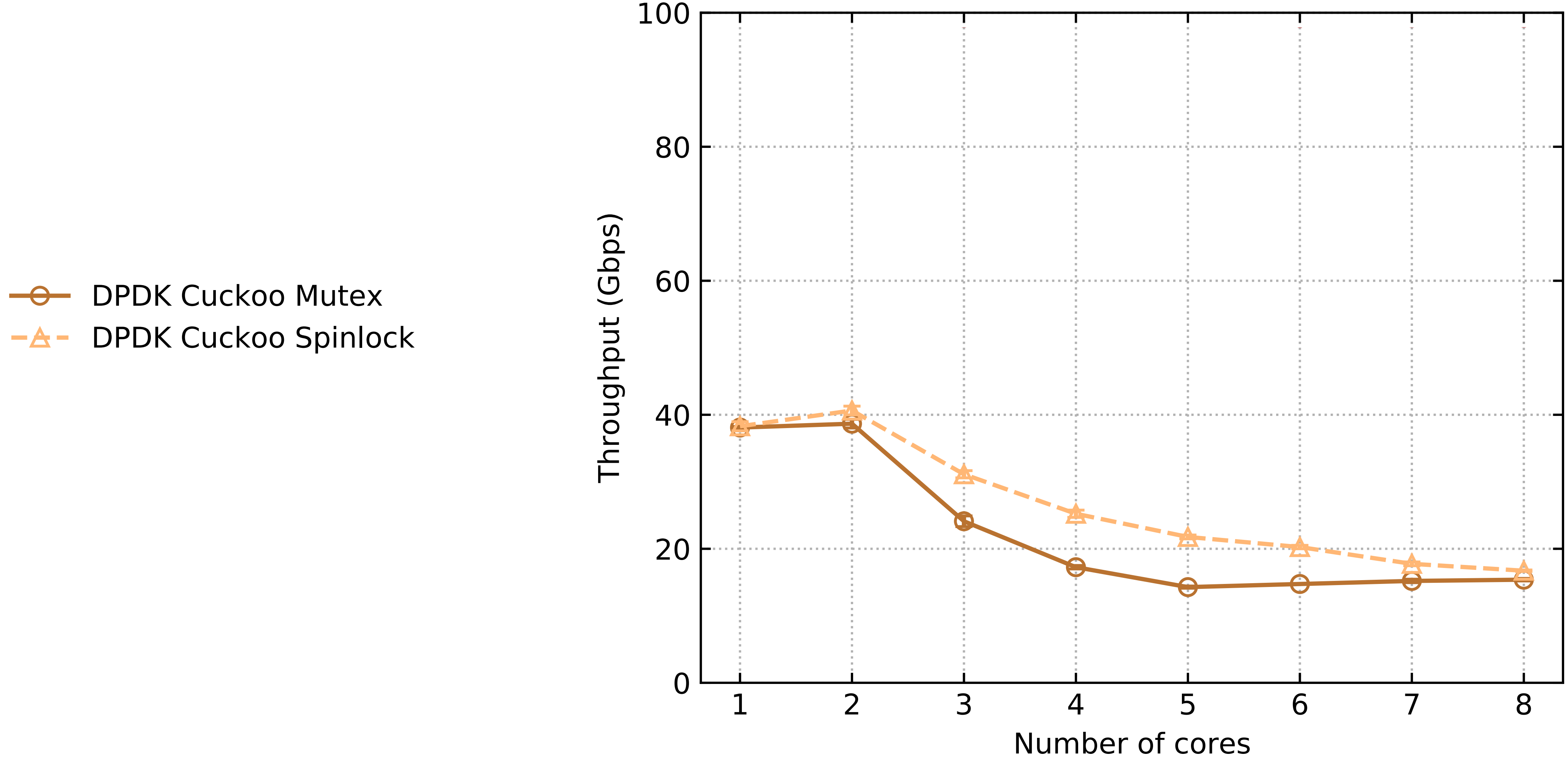
Core scaling: single table

Core scaling: single table



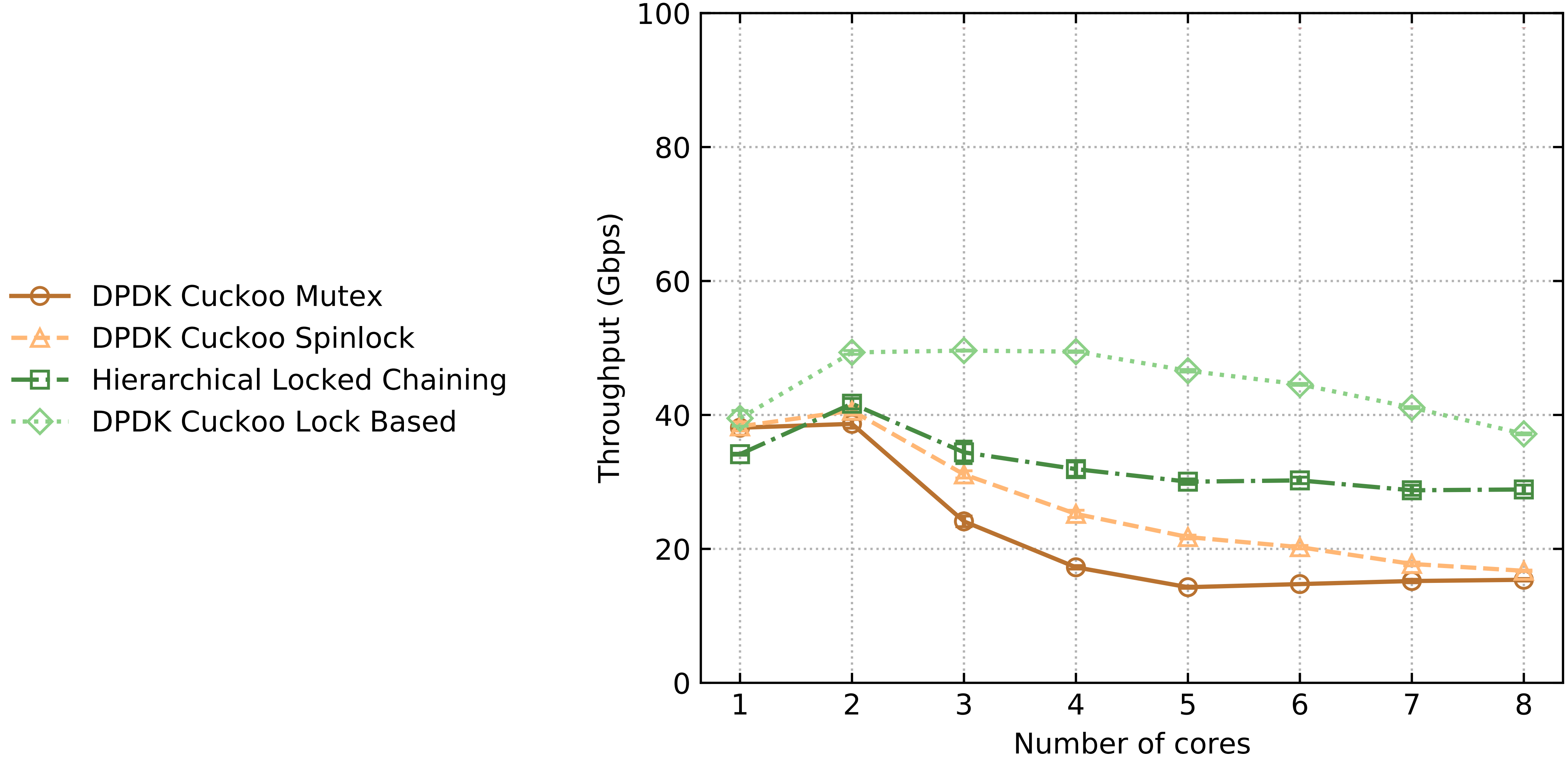
8M entries hash table, ≈ 100 Gbps

Core scaling: single table



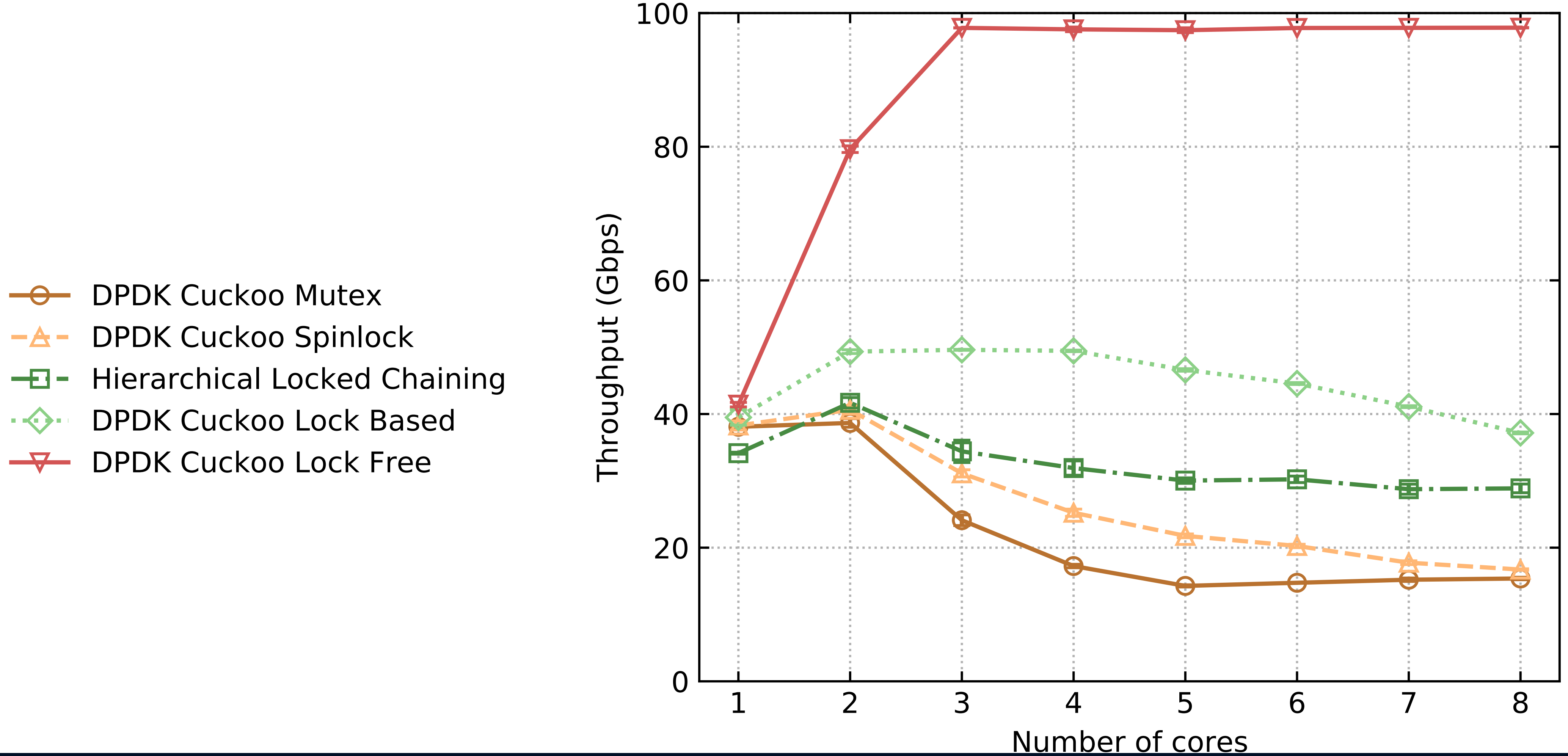
8M entries hash table, ≈ 100 Gbps

Core scaling: single table



8M entries hash table, ≈ 100 Gbps

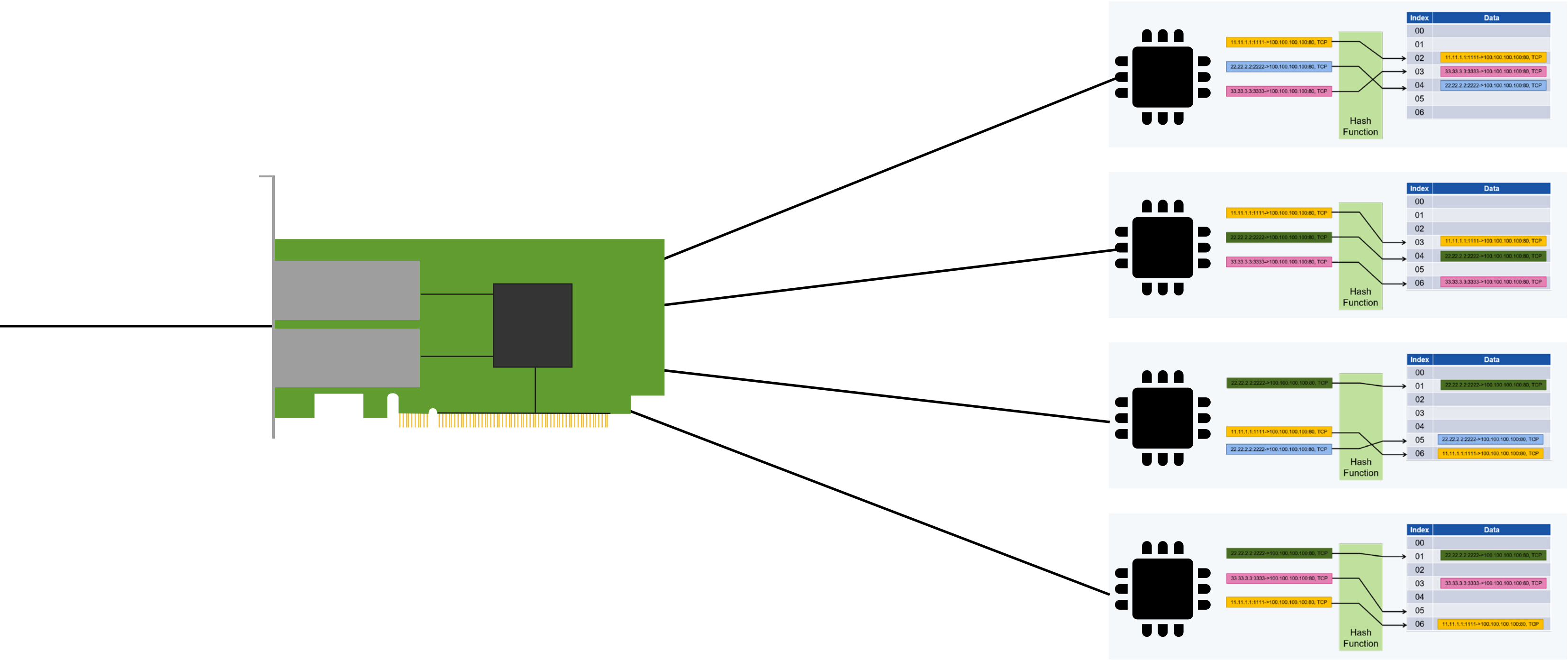
Core scaling: single table



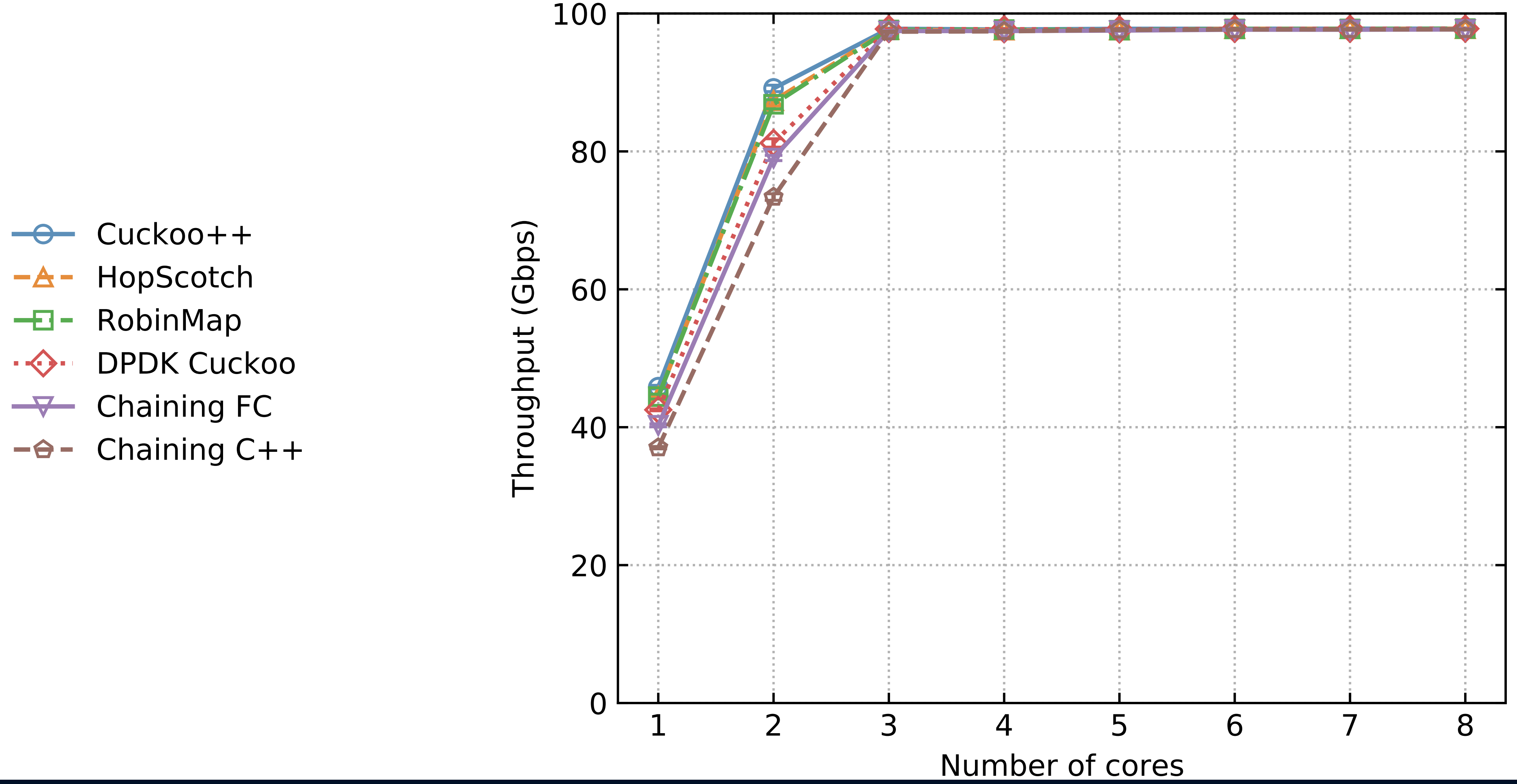
Lock-Free heavily depends on the workload

Multi core scaling: core sharding

Leverage RSS to spread packets to multiple independent cores



Core scaling: sharding



All implementations scale always linearly

The aging dilemma

When flows terminate, their entries should be removed

- Timer-based approach is needed
- Deletion could be more delicate than insertion: concurrency
- Three implementations studied:
  - Scanning
  - Lazy Deletion
  - Timing Wheels

Flow Table maintenance techniques

- **Scanning**

  Parse the table periodically, deleting expired entries.

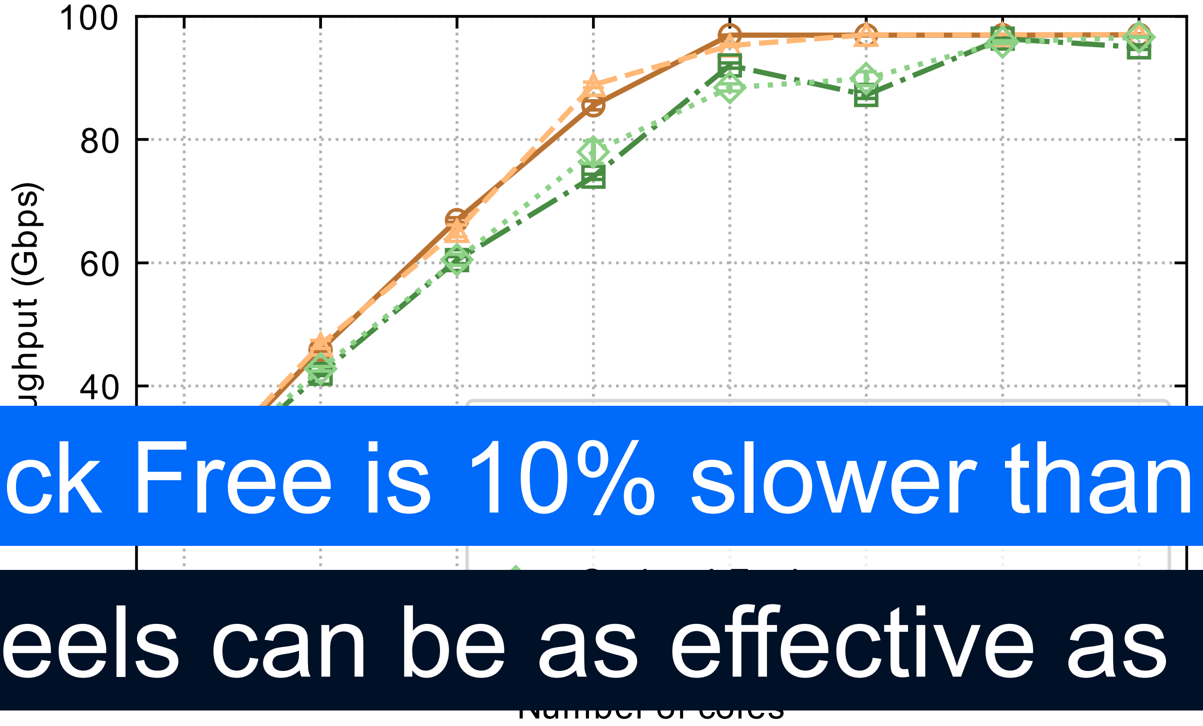
- **Lazy Deletion**

  compare last access time upon collisions.

- **Timing Wheels**

  entries are registered in time-based buckets.

Deletion: scaling



At scale, Lock Free is 10% slower than Core Sharding

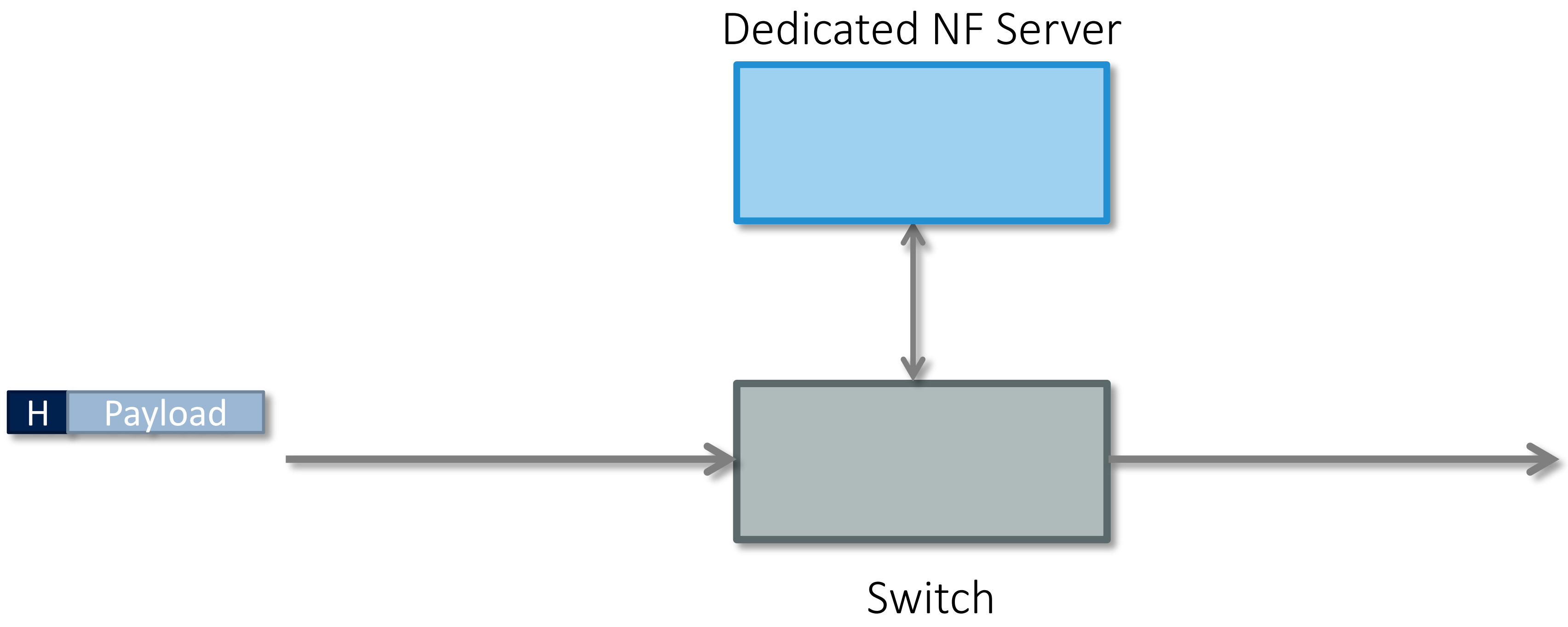
Timing wheels can be as effective as Lazy Deletion

Caida, 4M hash tables, 32x parallel traces ≈ 100 Gbps

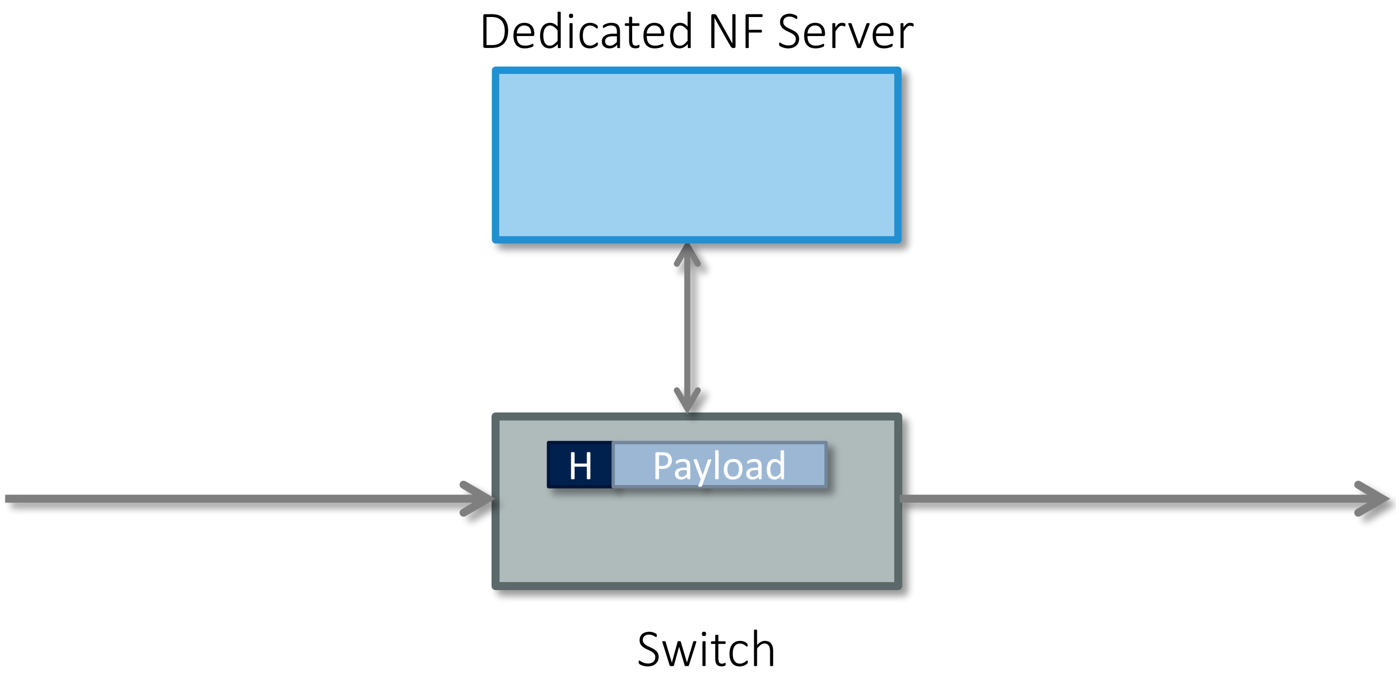
Conclusion: we have spare capacity

Can we design a packet processing pipeline that handles
one terabit per second of traffic
on a single dedicated device?

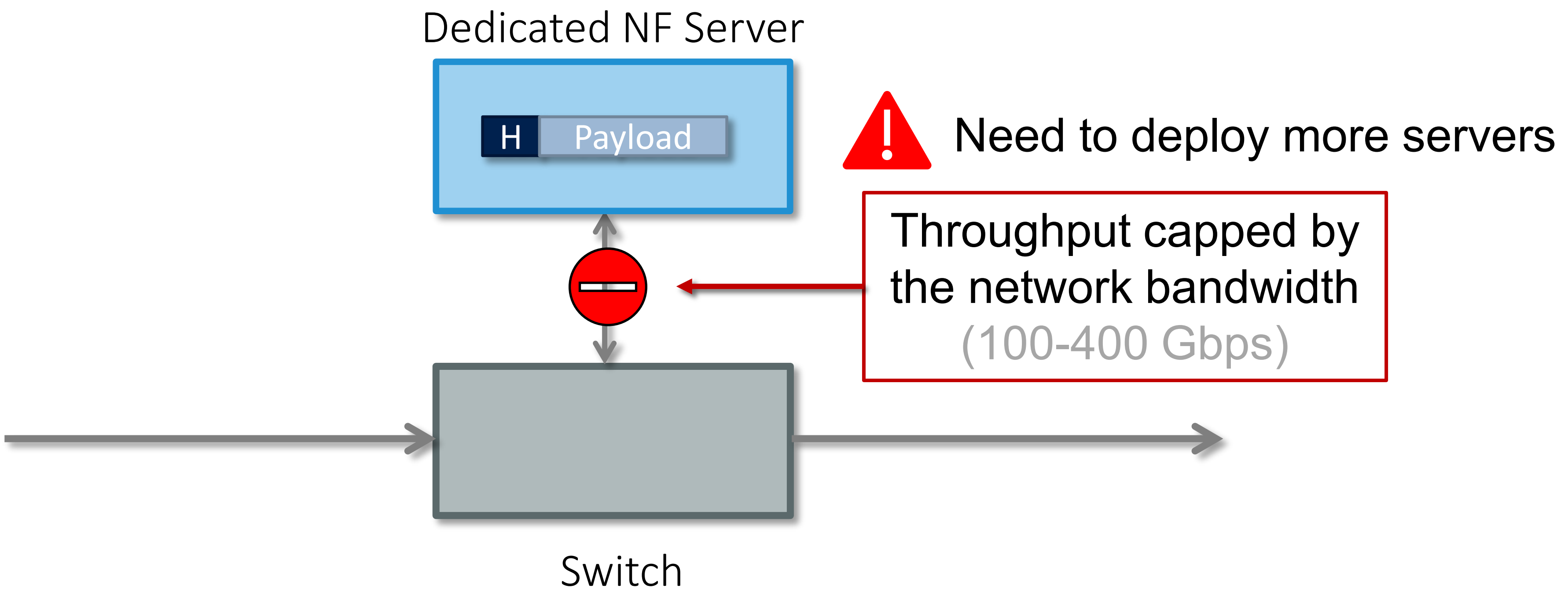
The bandwidth limit



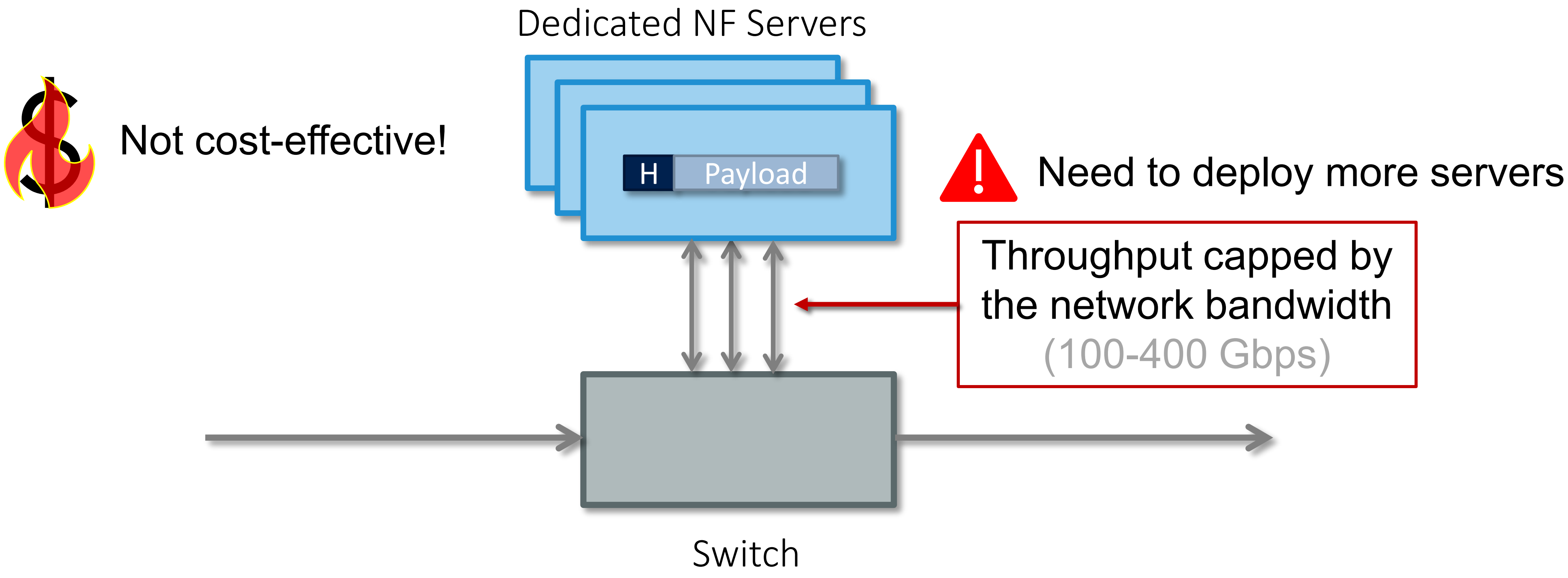
The bandwidth limit



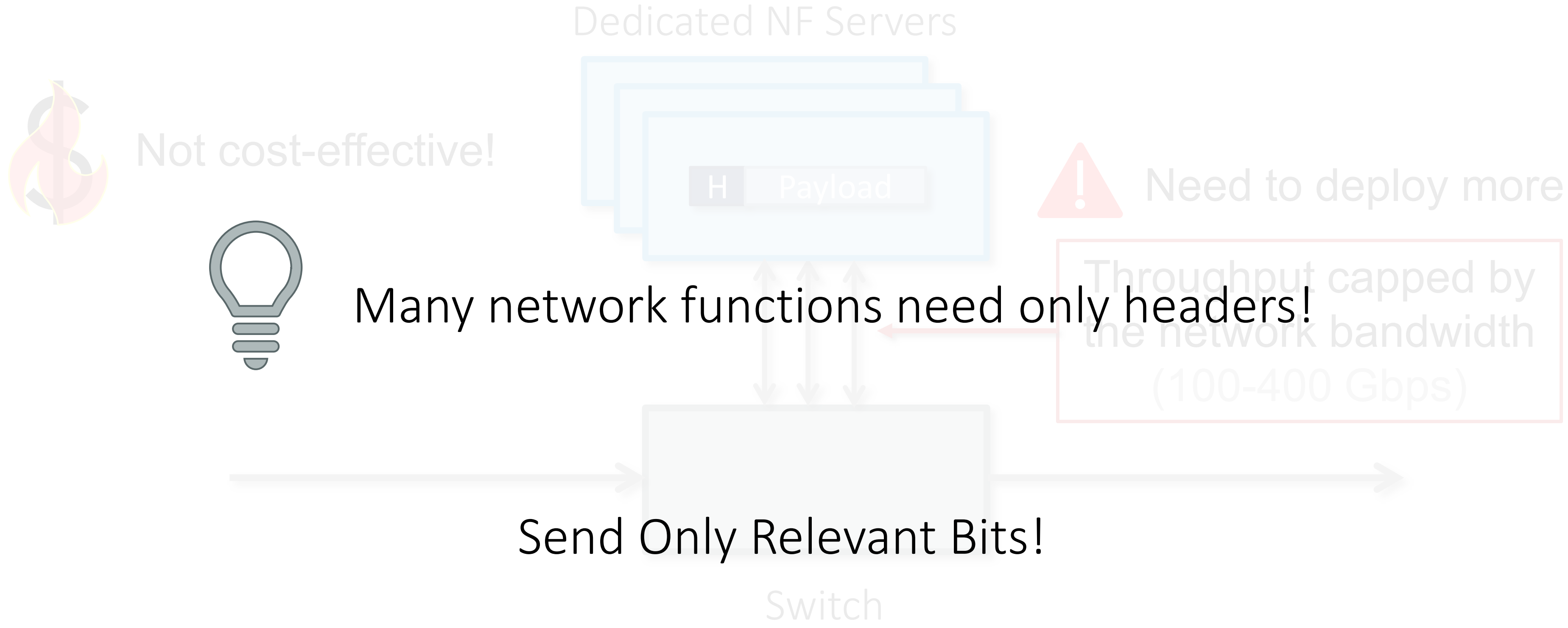
The bandwidth limit



The bandwidth limit



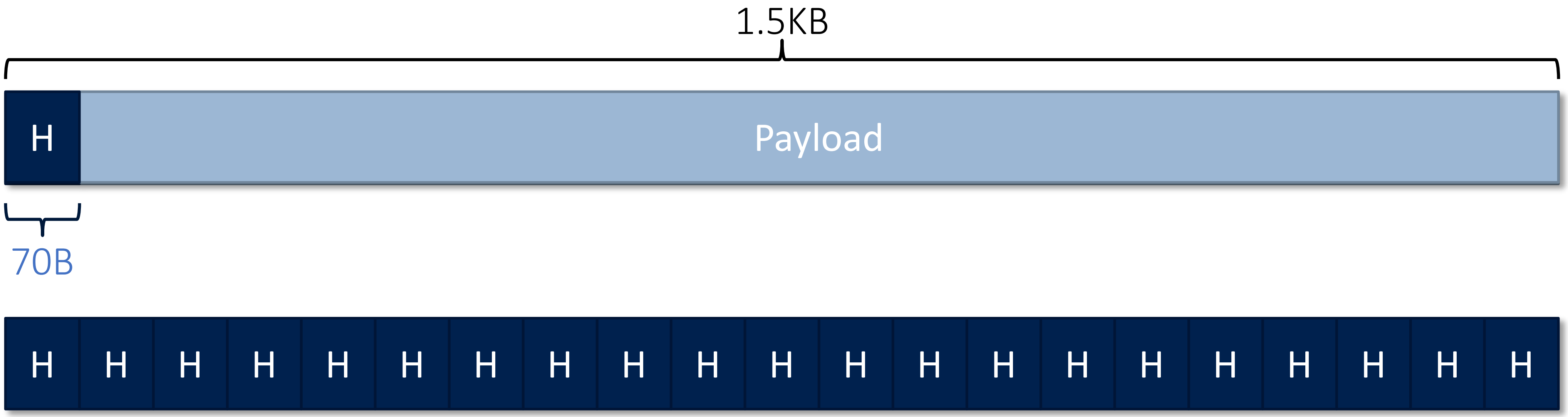
The bandwidth limit



Many network functions need only headers!

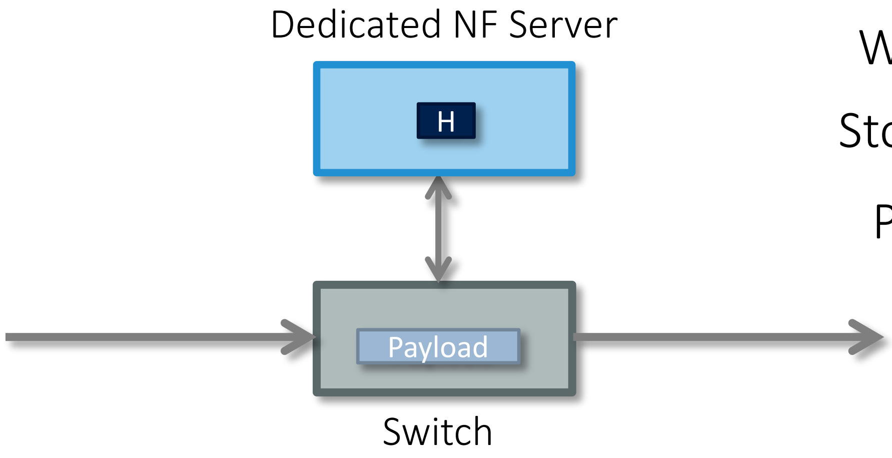
Send Only Relevant Bits!

Send Only Relevant Bits!



- Free up bandwidth
- Higher cache-hit ratio

Send Only Relevant Bits!

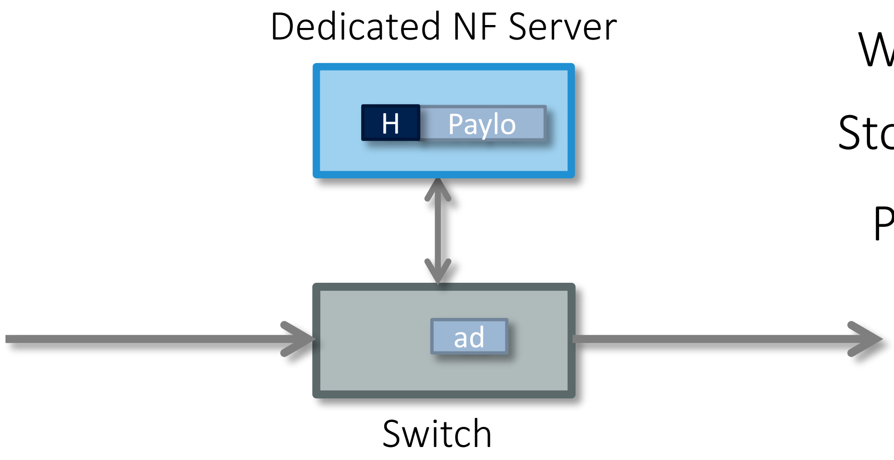


Where to store payloads?

Store Payloads on the Switch

PayloadPark [CoNEXT '20]

Send Only Relevant Bits!



Where to store payloads?

Store Payloads on the Switch

PayloadPark [CoNEXT '20]

Store Payloads on the Switch

What is the impact?

Store Payloads on the Switch

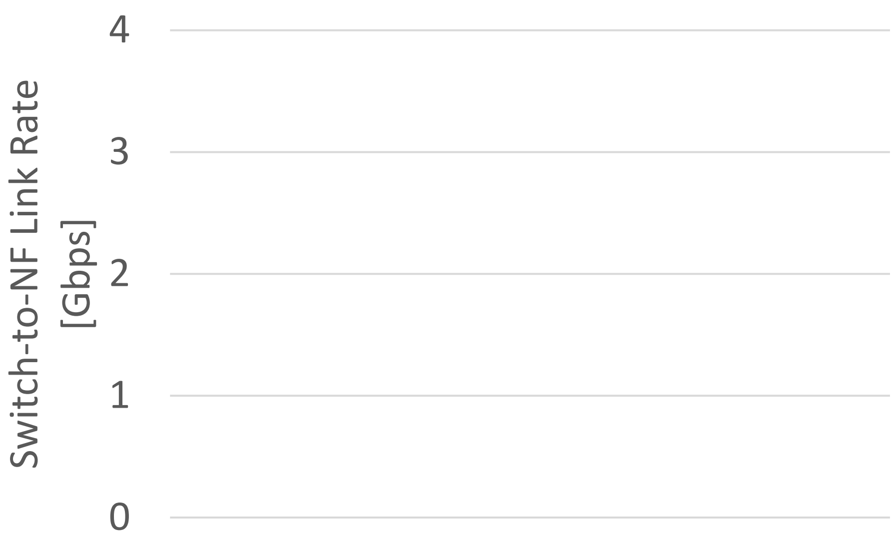
What is the impact?

Let's examine a CAIDA trace

Store Payloads on the Switch

What is the impact?

Let’s examine a CAIDA trace

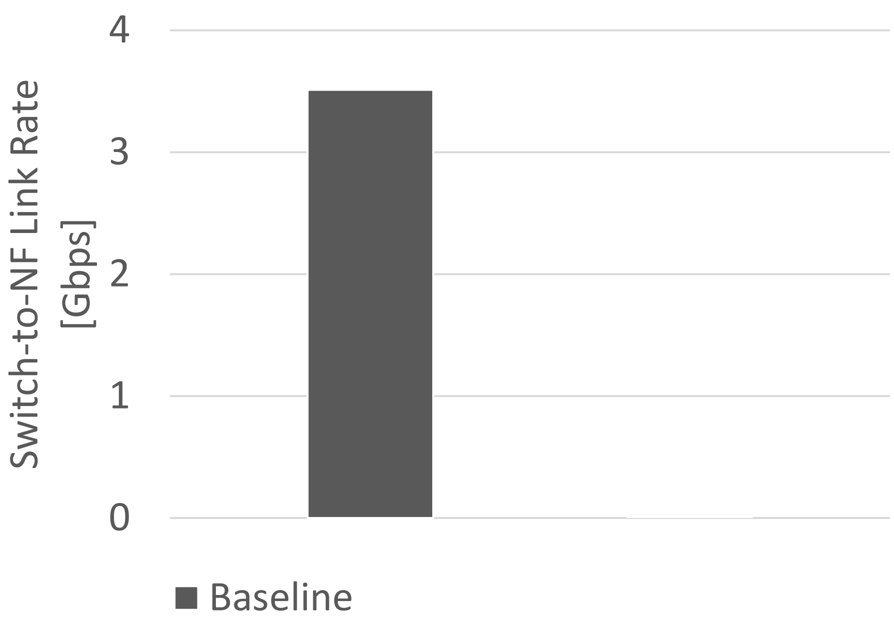


Packets sent to the NF

Store Payloads on the Switch

What is the impact?

Let's examine a CAIDA trace

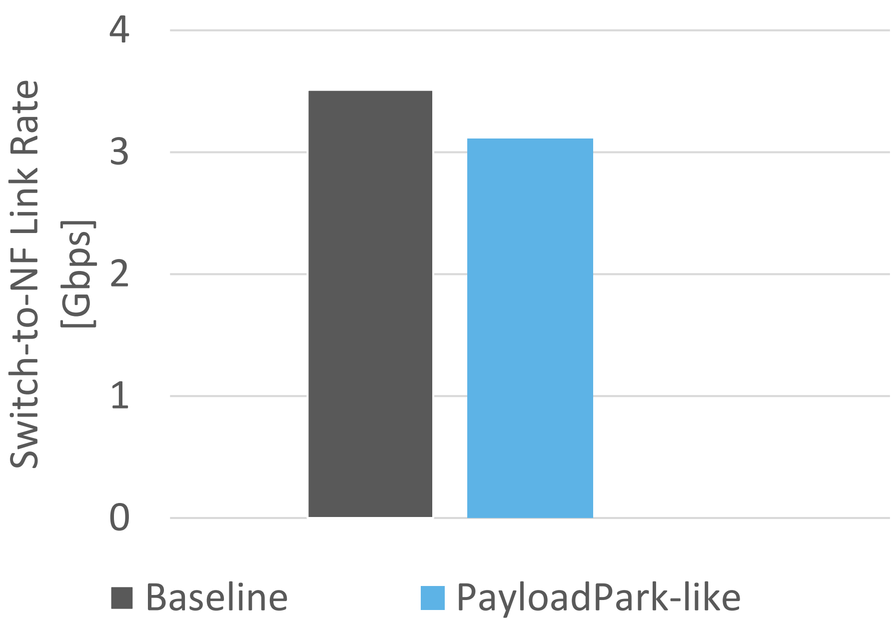


Packets sent to the NF

Store Payloads on the Switch

What is the impact?

Let's examine a CAIDA trace

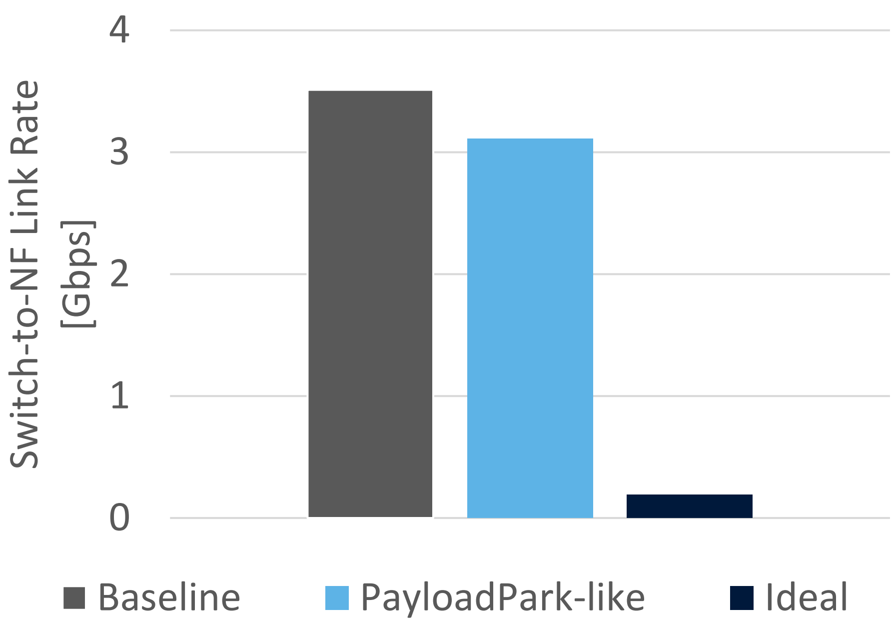


Packets sent to the NF

Store Payloads on the Switch

What is the impact?

Let's examine a CAIDA trace

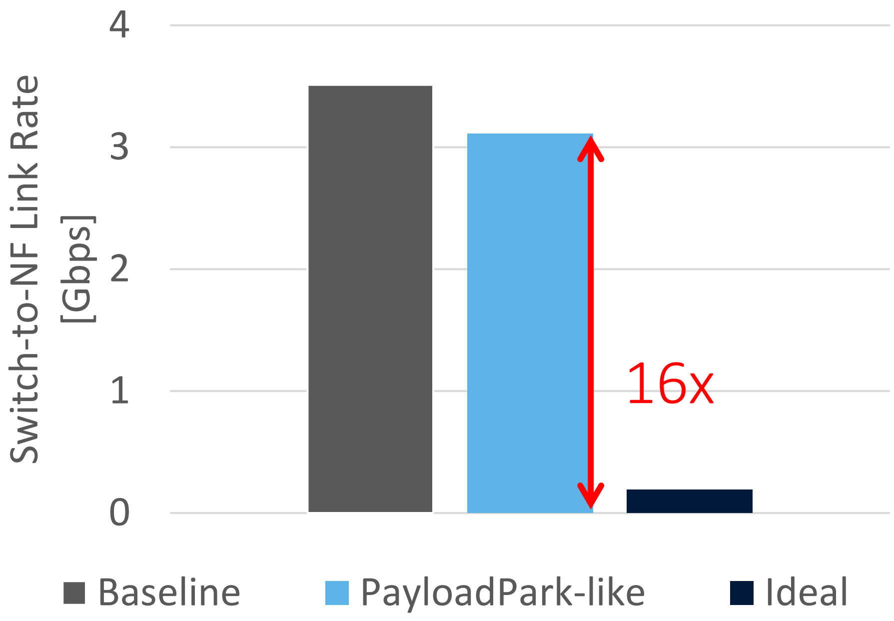


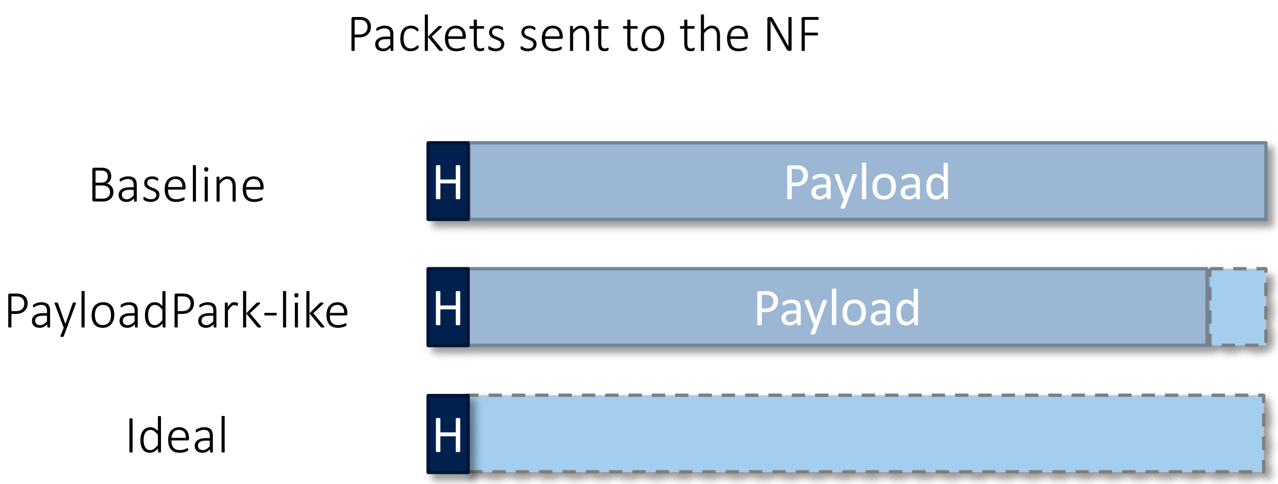
Packets sent to the NF

Store Payloads on the Switch

What is the impact?

Let’s examine a CAIDA trace

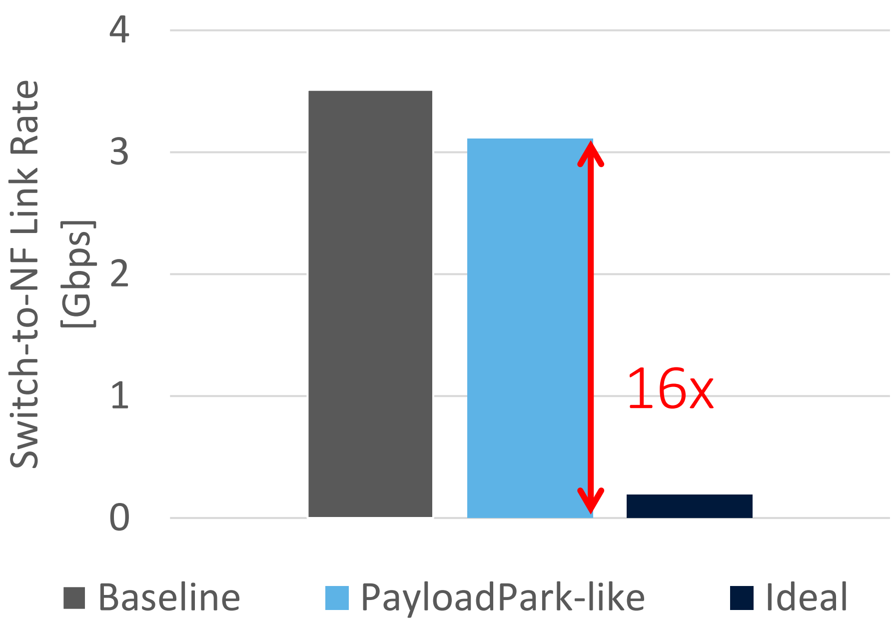


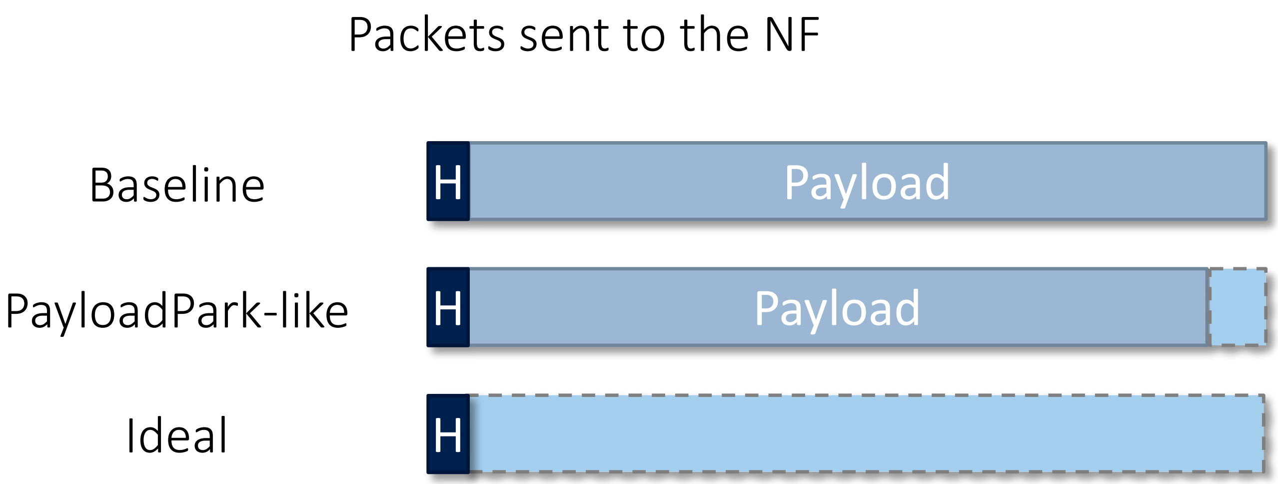


Store Payloads on the Switch

What is the impact?

Let's examine a CAIDA trace

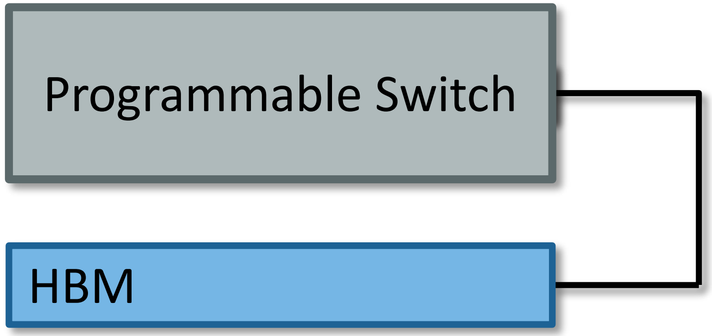




How to extend the switch memory?

How to extend the switch memory?

- Using a dedicated external memory (*e.g.*, HBM)
  - ✓ Simple solution

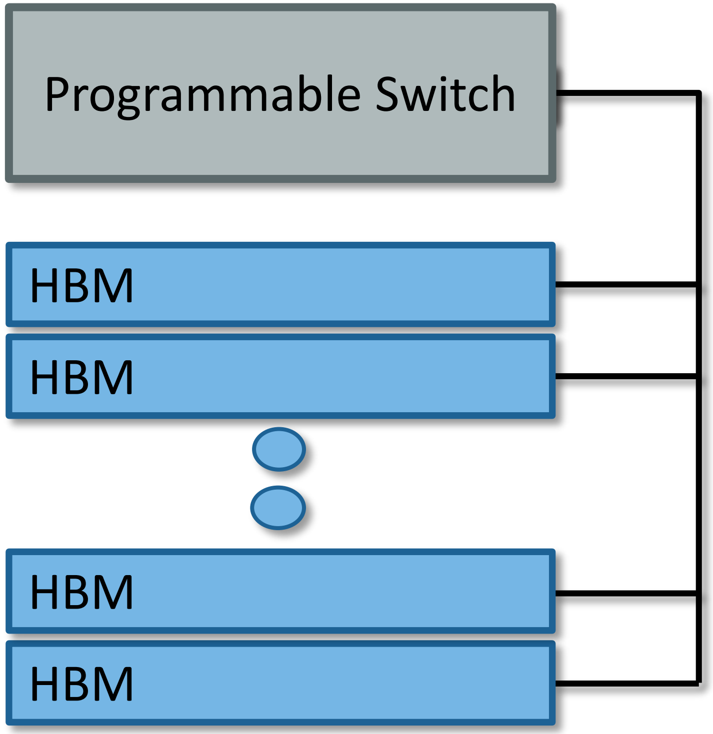


How to extend the switch memory?

❌ Not cost-effective

Using a dedicated external memory (*e.g.*, HBM)

- ✓ Simple solution
- ⬇ Higher energy footprint
- ⬇ High-cost
- ! Wastes some ports on the switch

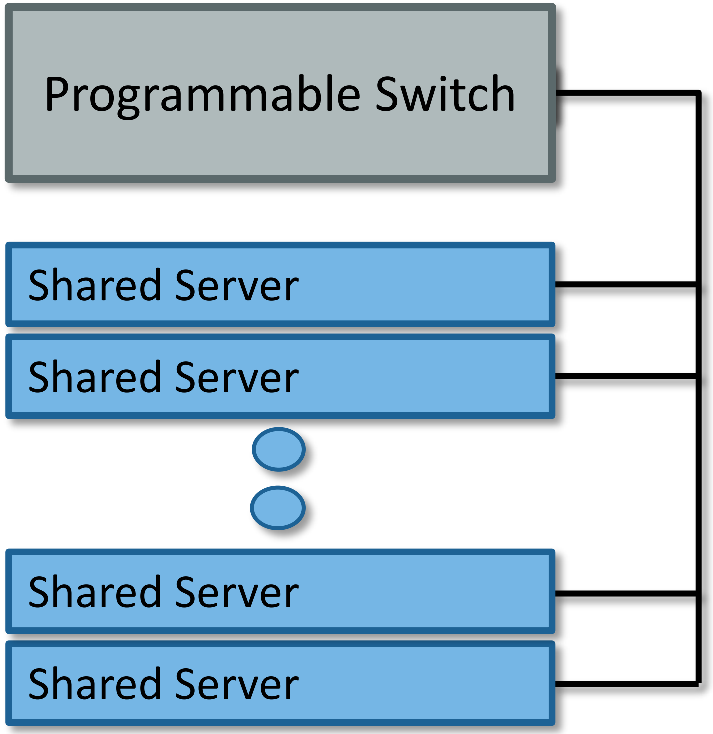


How to extend the switch memory?

The Ribosome Approach

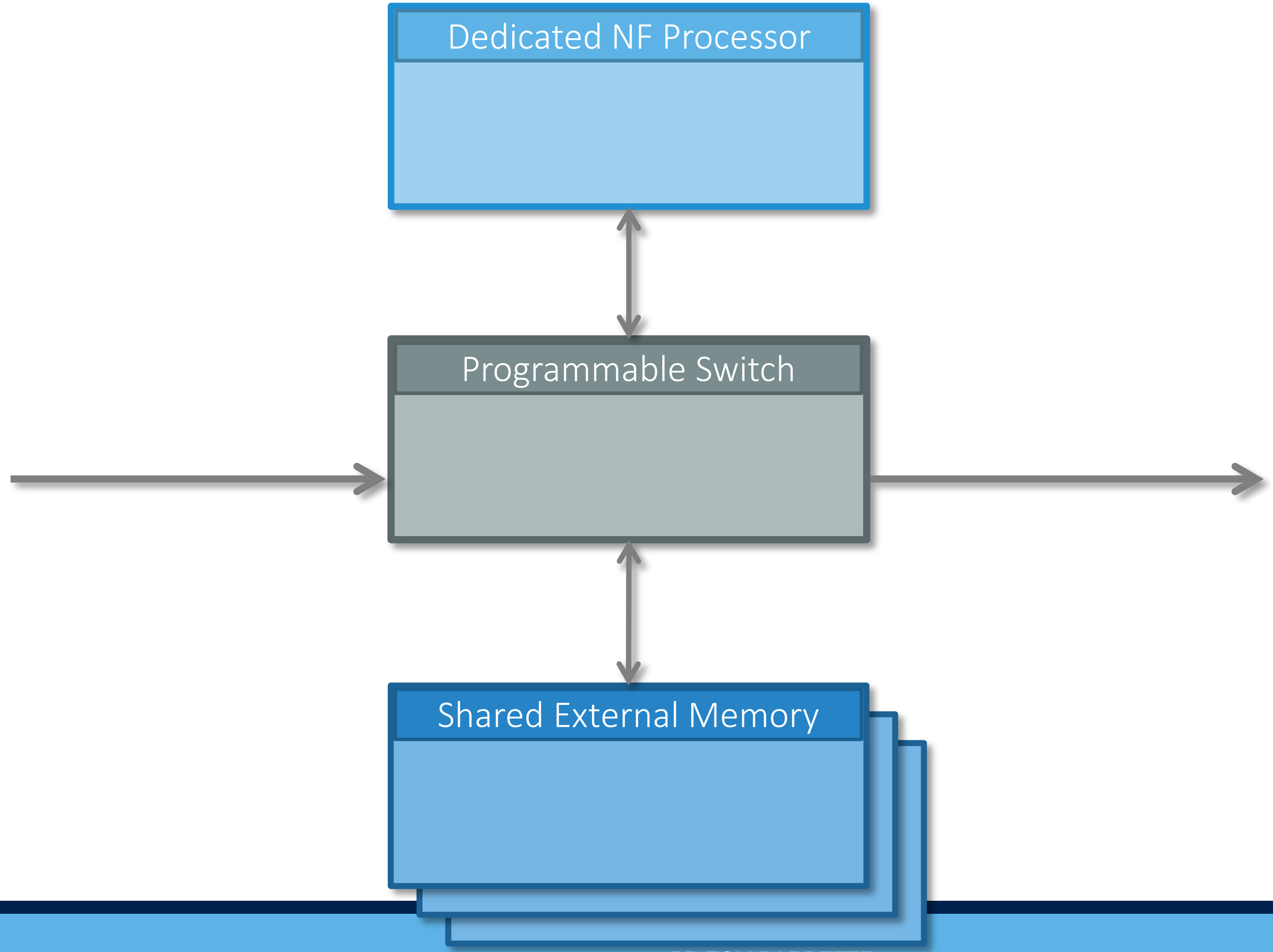
Exploiting a disaggregated pipeline on shared servers

- Many spare resources in the datacenter
- Better resources usage
- Low-cost

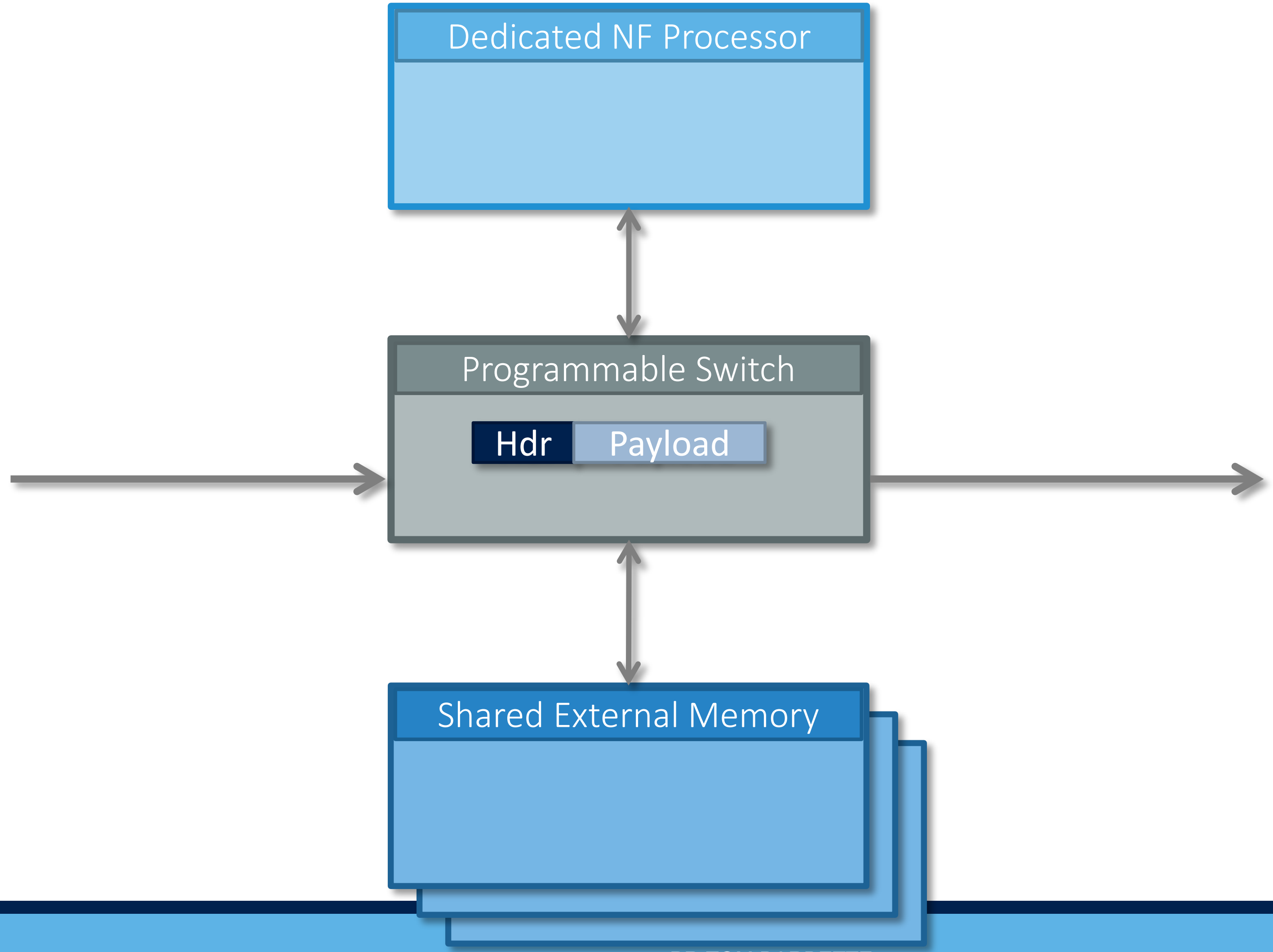


[NSDI'23] Mariano Scazzariello, Tommaso Caiazzi, Hamid Ghasemirahni, Tom Barbette, Dejan Kostić, Marco Chiesa

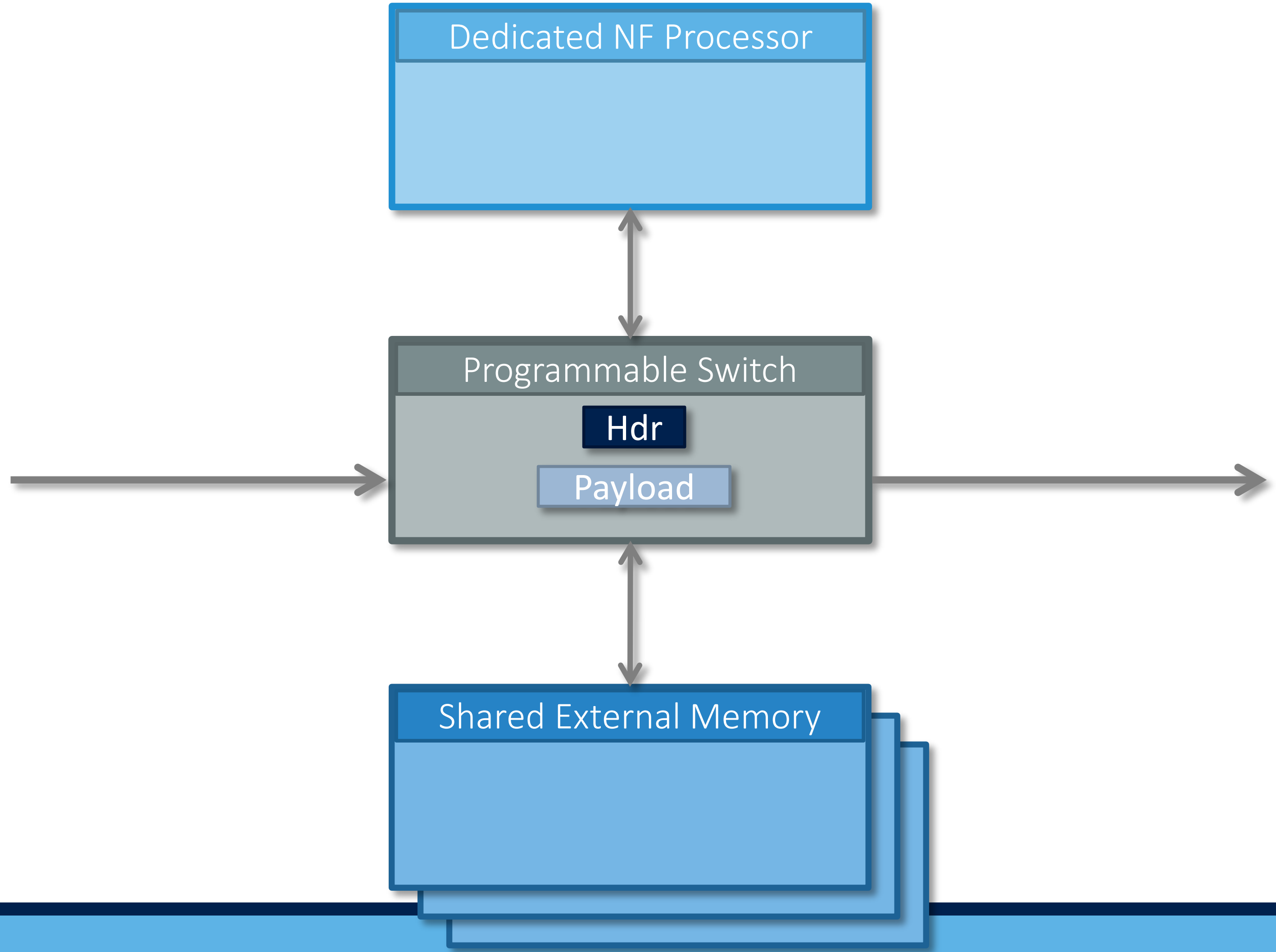
The Ribosome Approach



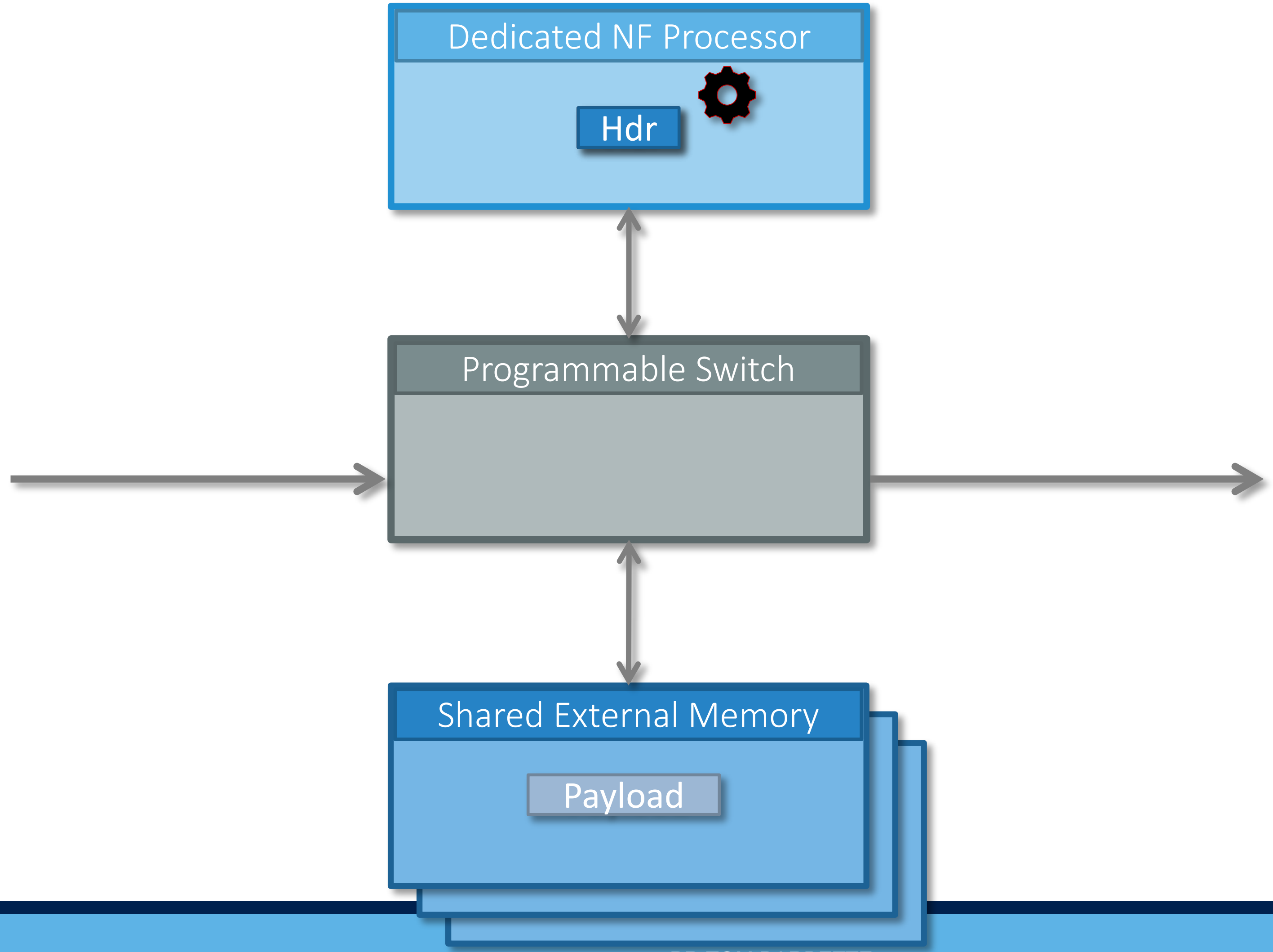
The Ribosome Approach



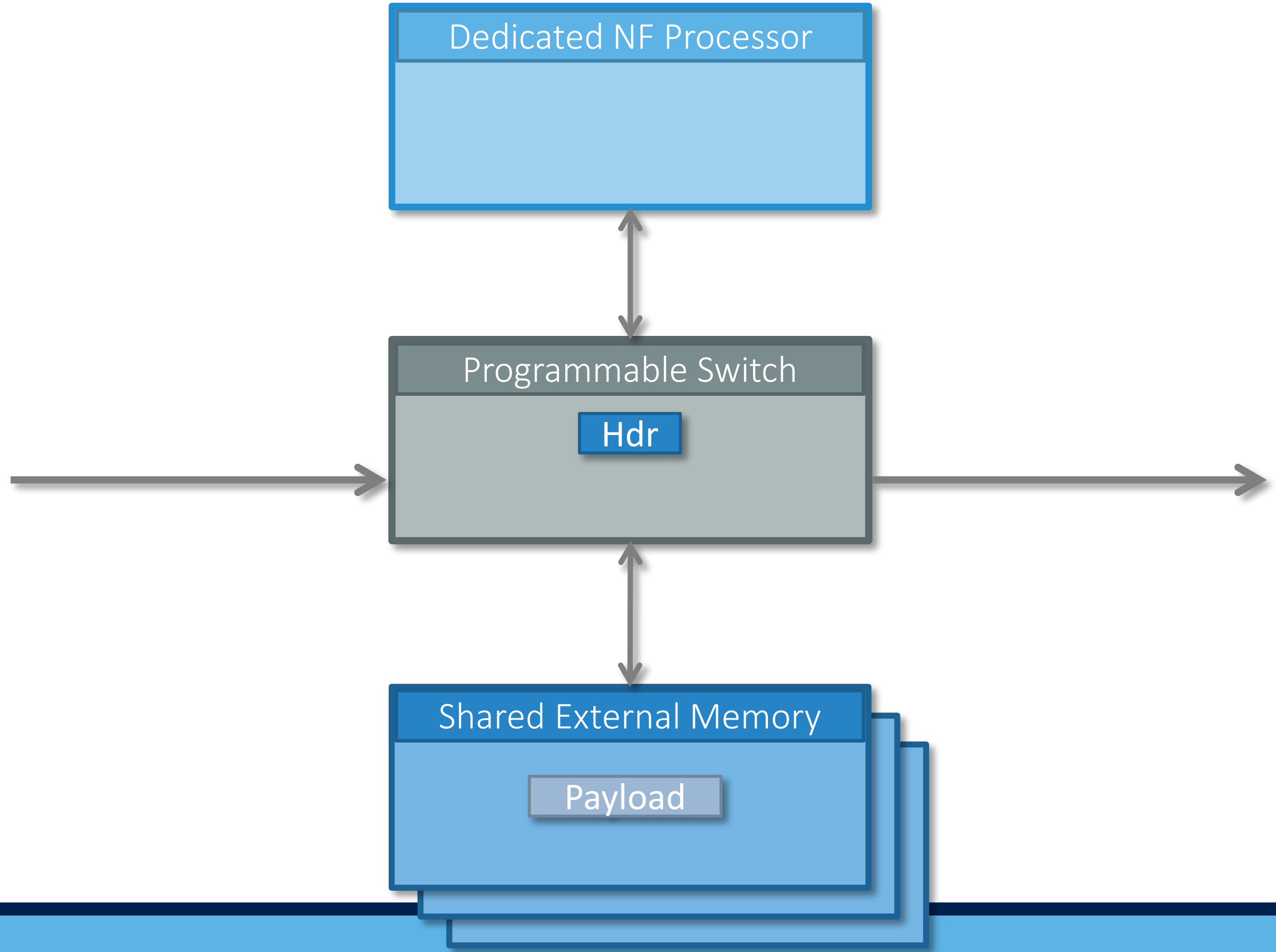
The Ribosome Approach



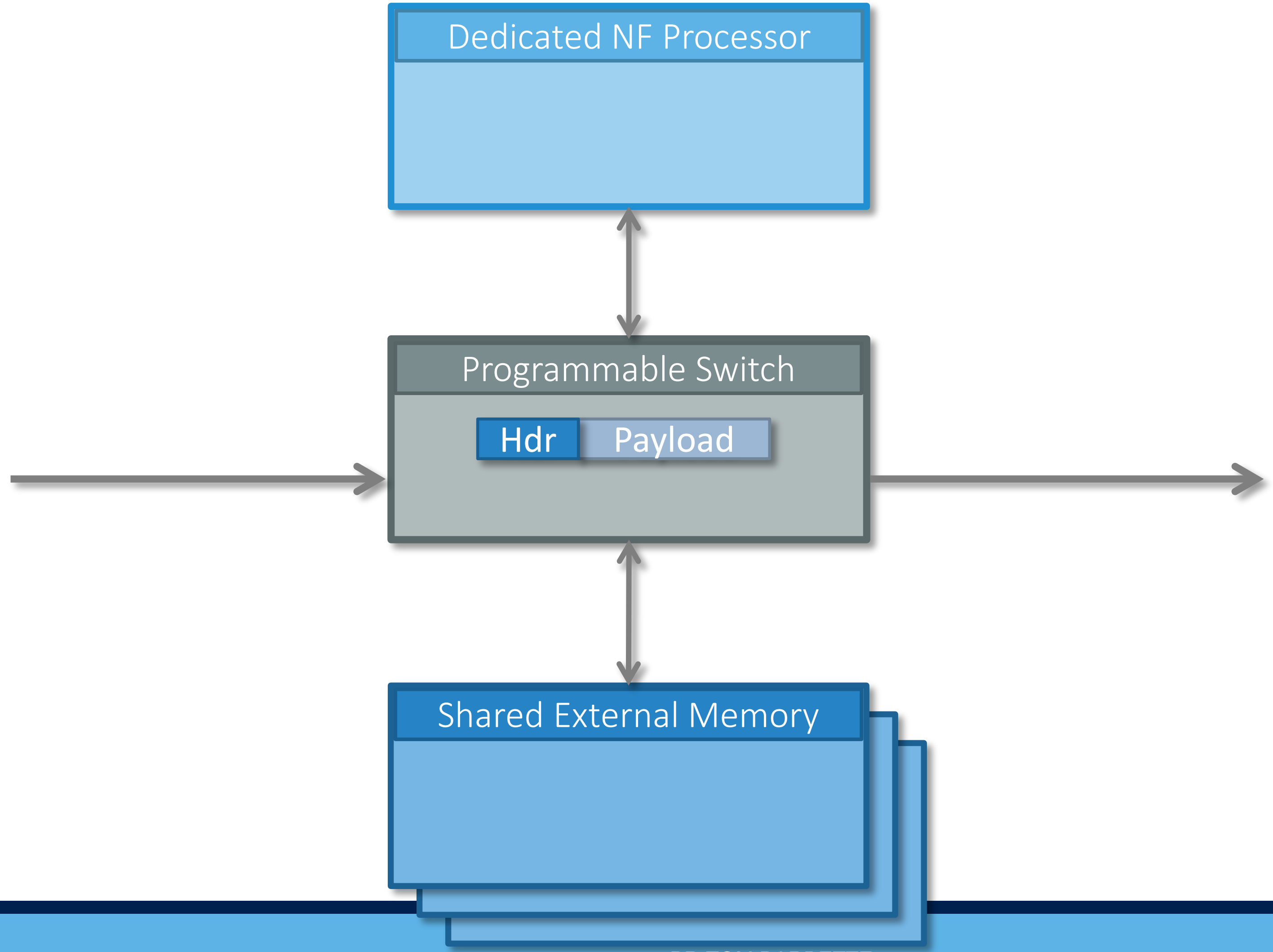
The Ribosome Approach



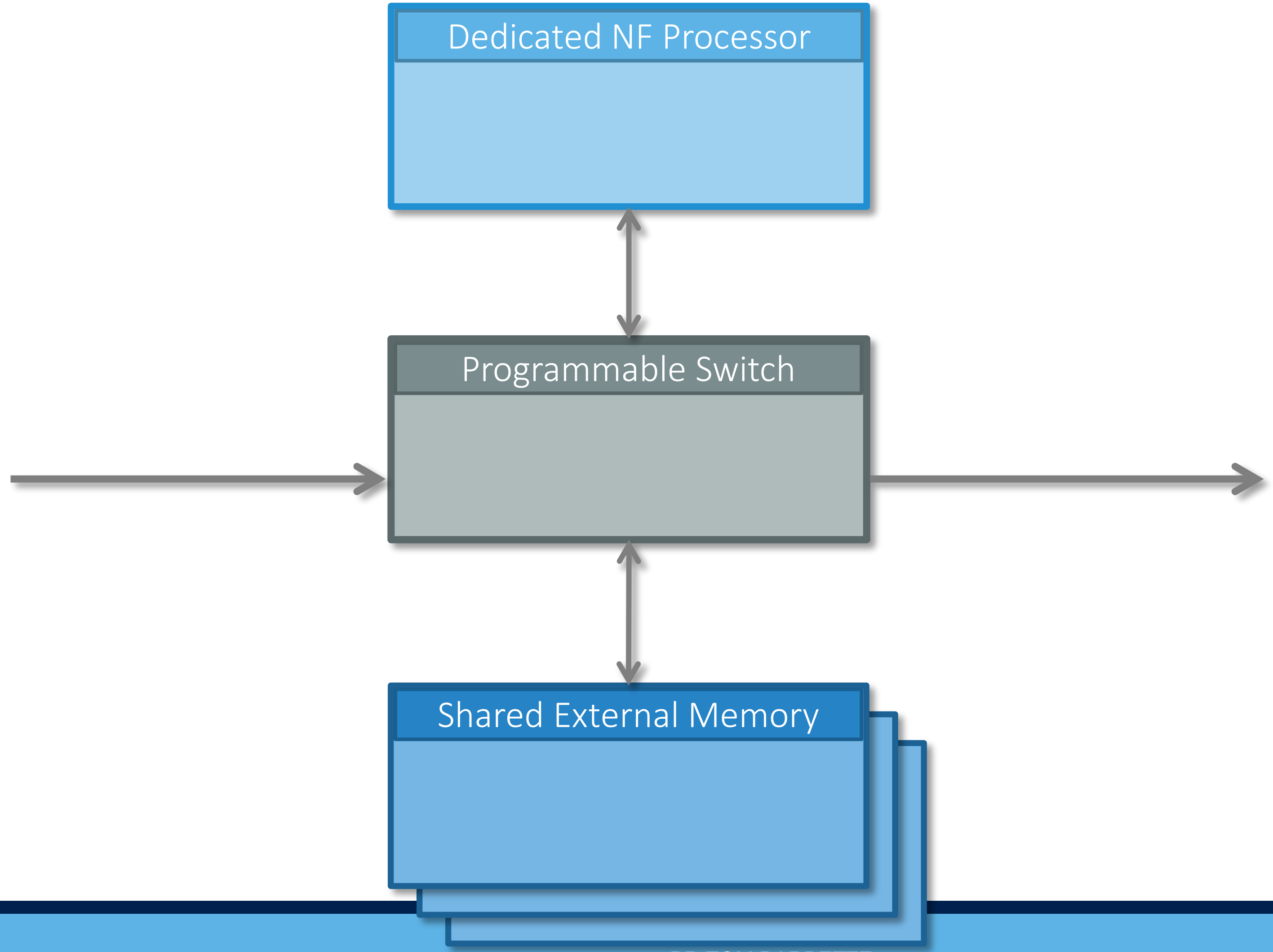
The Ribosome Approach



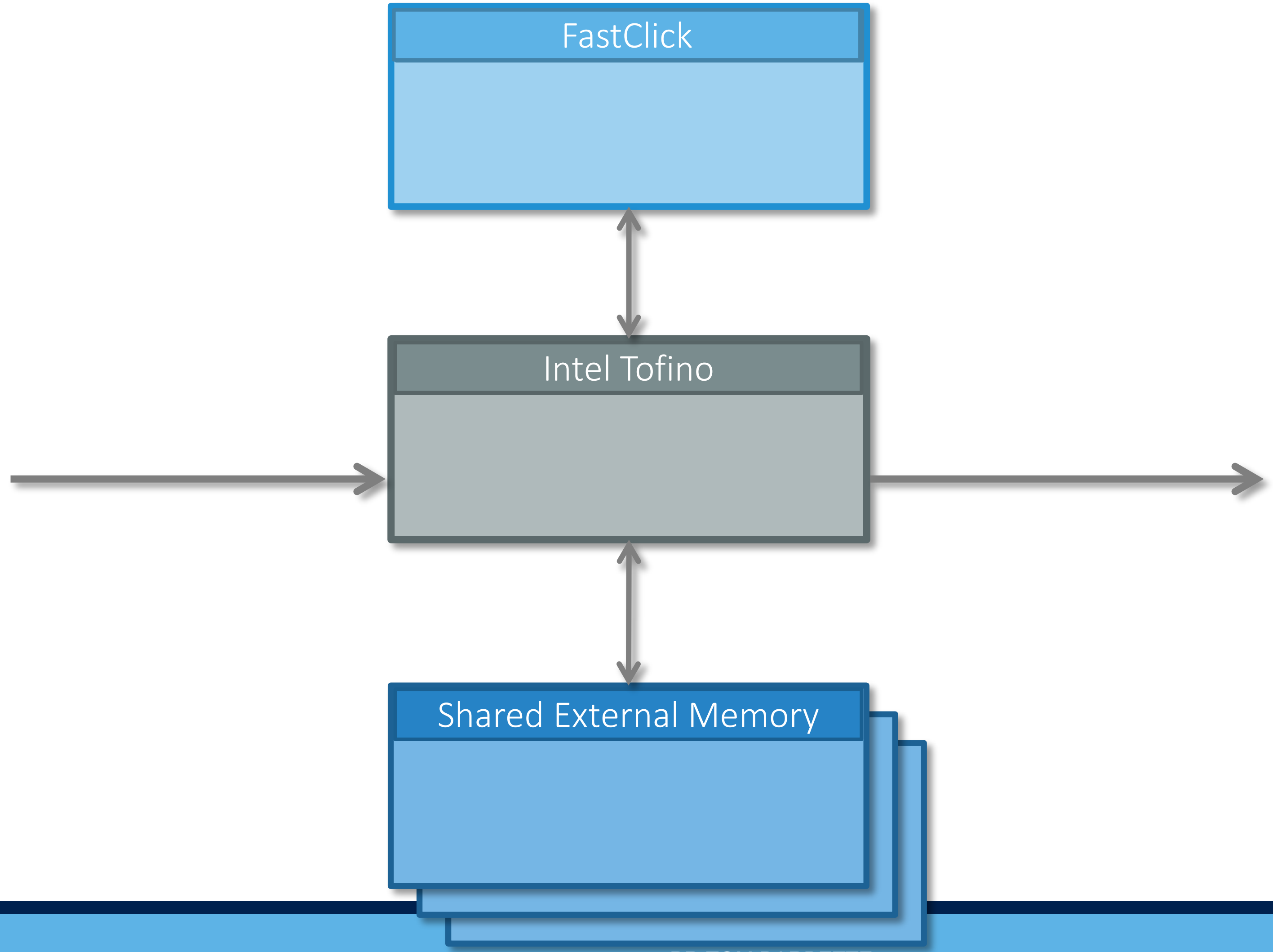
The Ribosome Approach



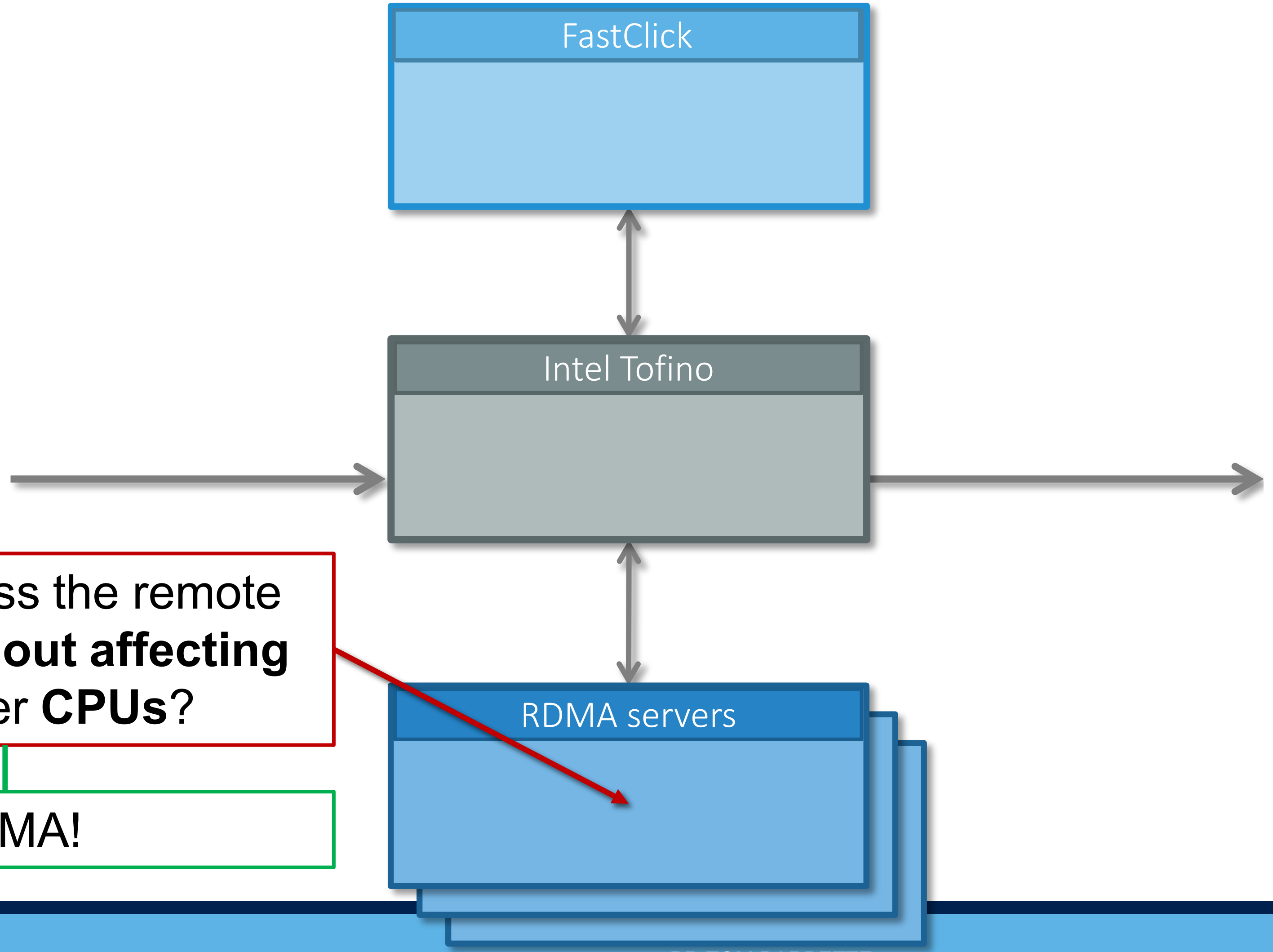
The Ribosome Approach



Implementation



Implementation



How to access the remote memory **without affecting** the server **CPUs**?

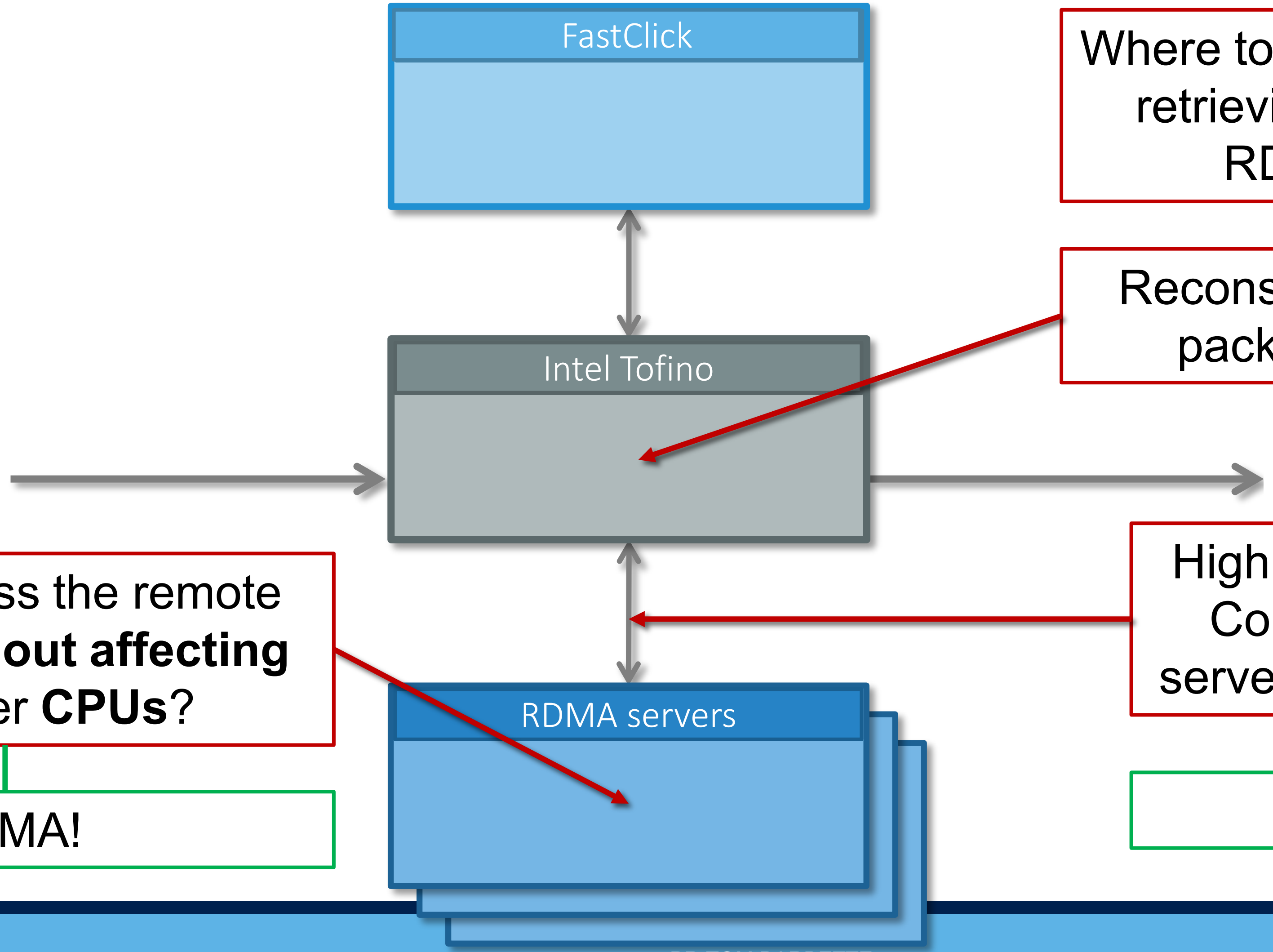
RDMA!

Challenges

Store Headers on the Switch

Where to store headers while retrieving payloads from RDMA servers?

Reconstructing millions of packets per second!



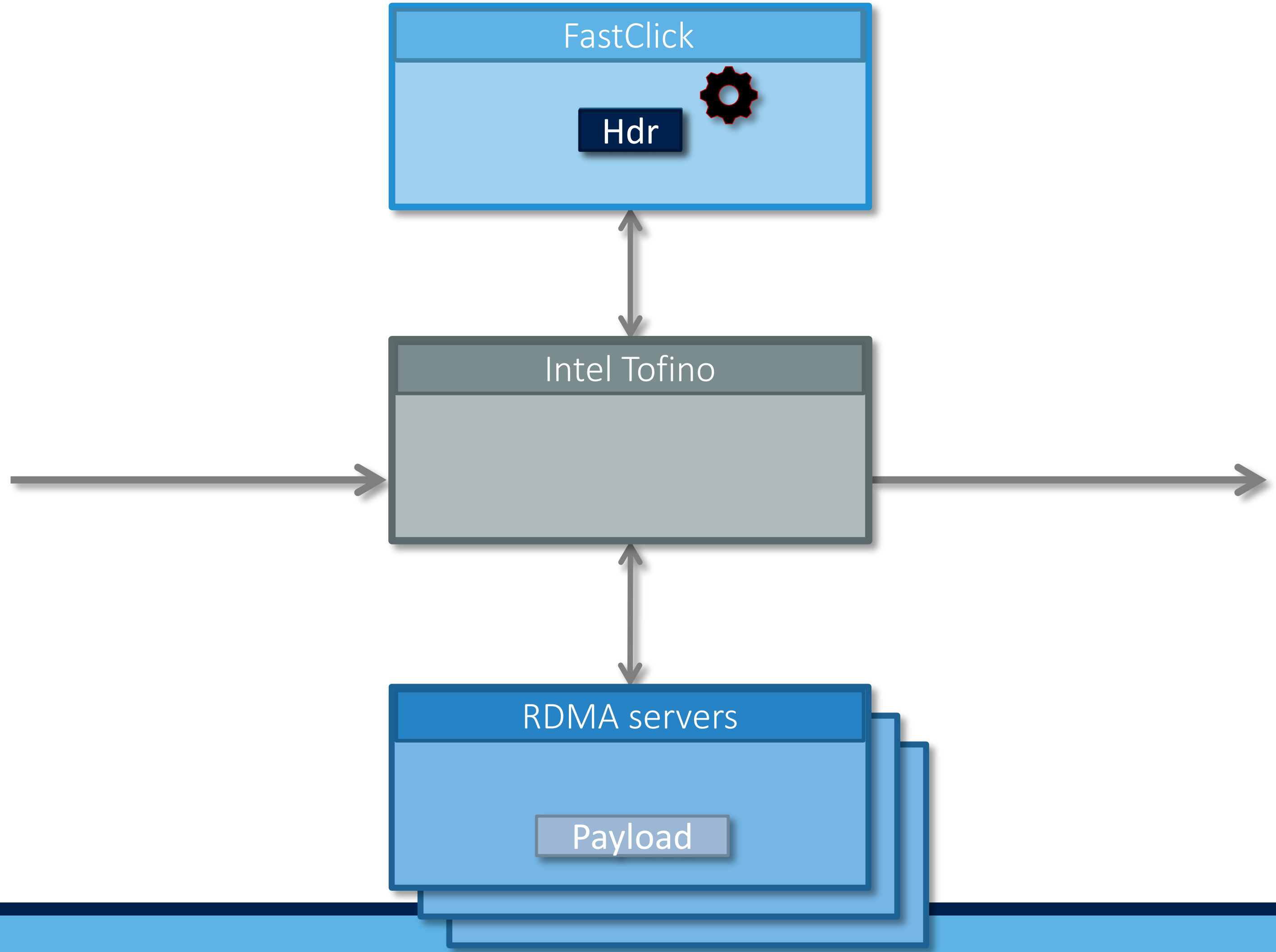
How to access the remote memory **without affecting** the server **CPUs**?

RDMA!

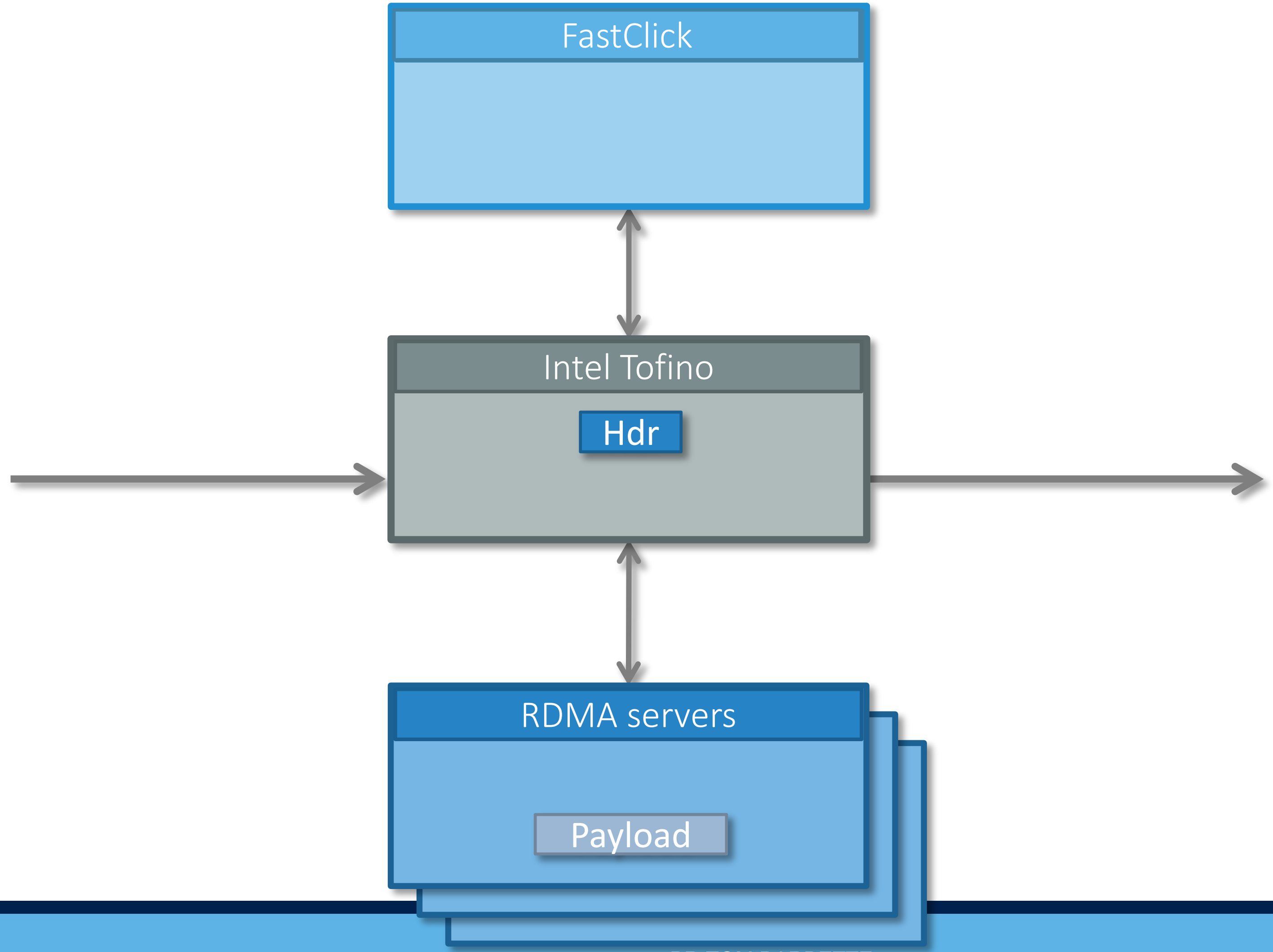
High speed reliable RDMA Connection on multiple servers for storing payloads

See the paper!

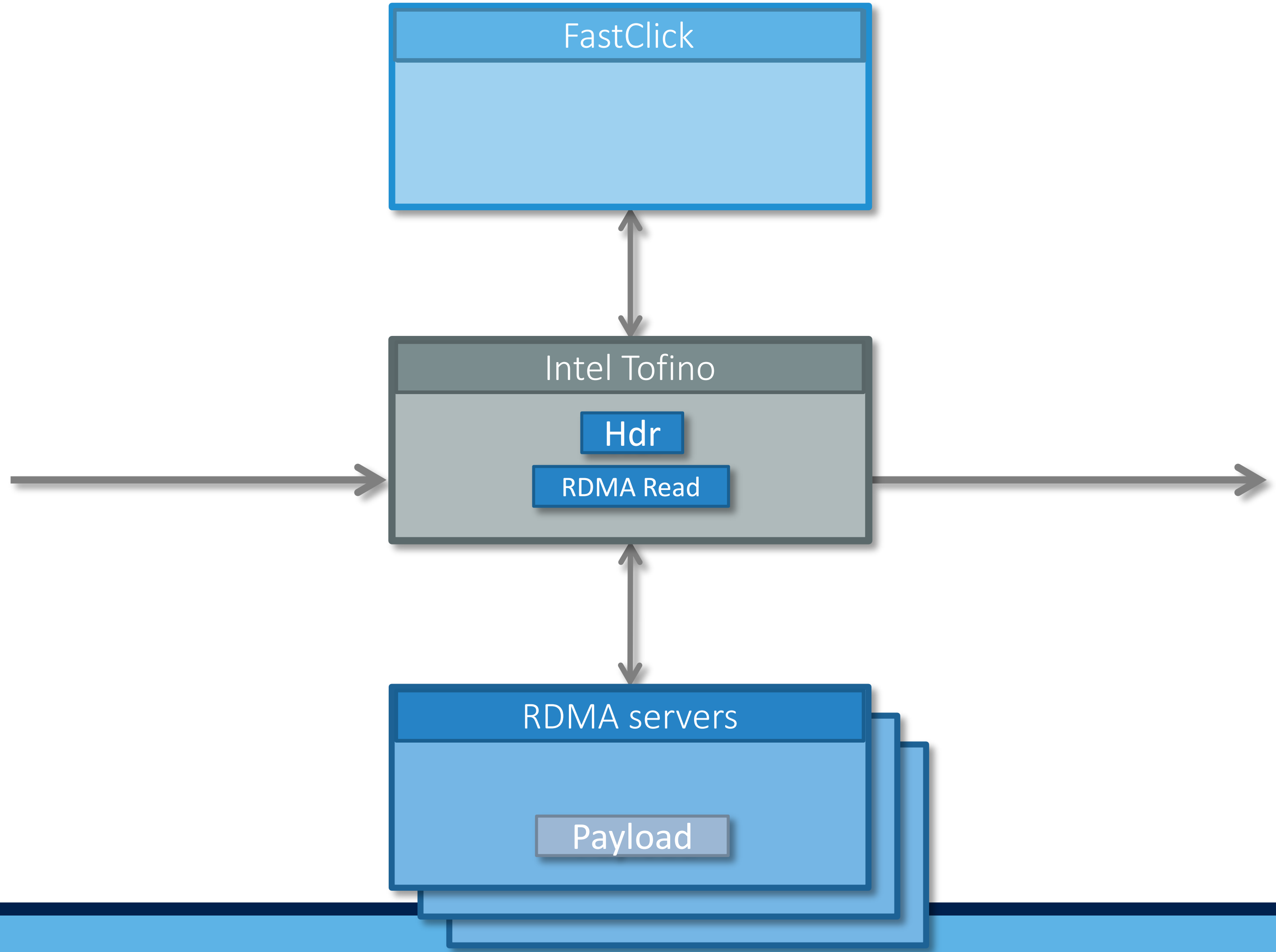
Store Headers on the Switch



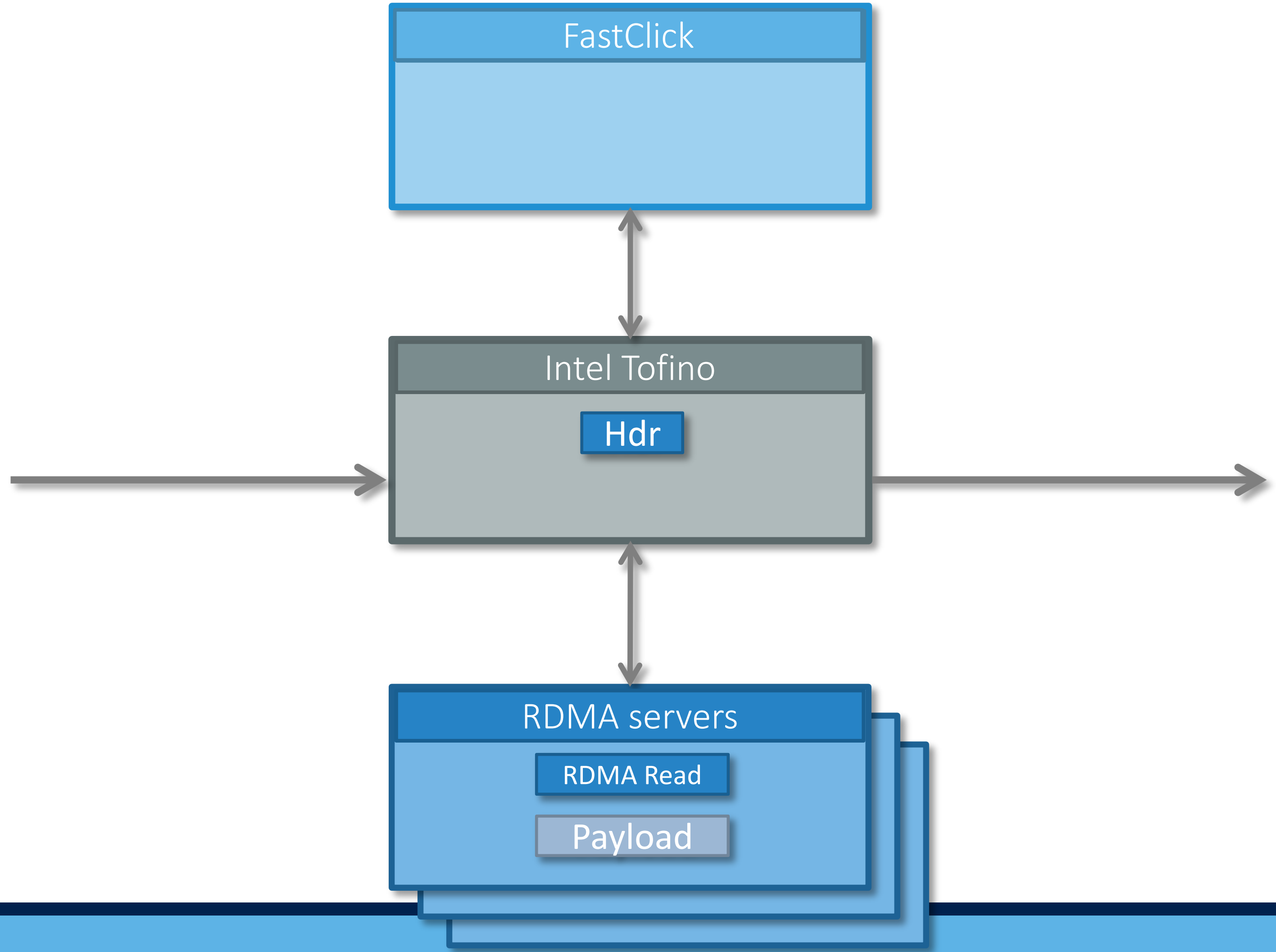
Store Headers on the Switch



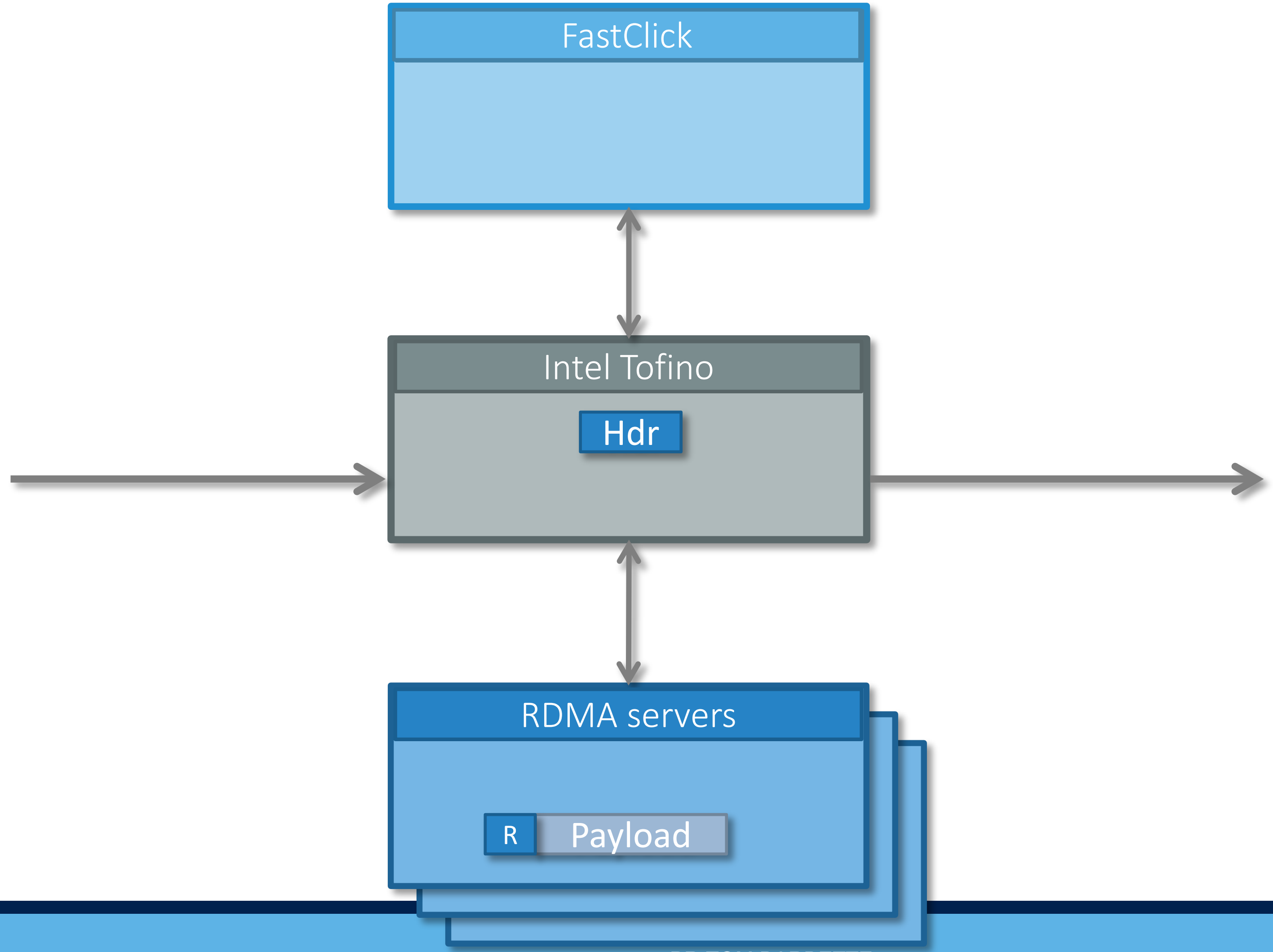
Store Headers on the Switch



Store Headers on the Switch

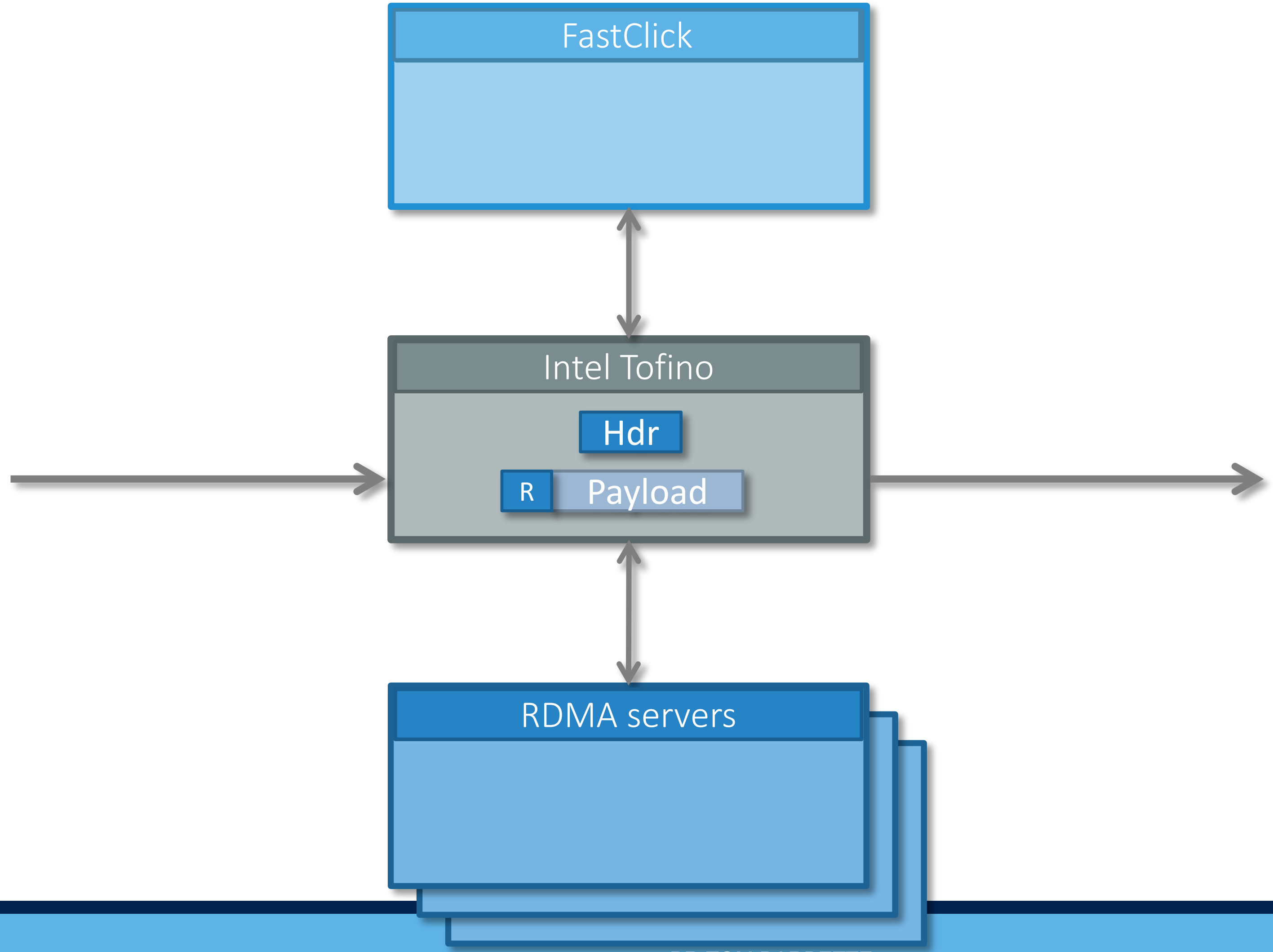


Store Headers on the Switch

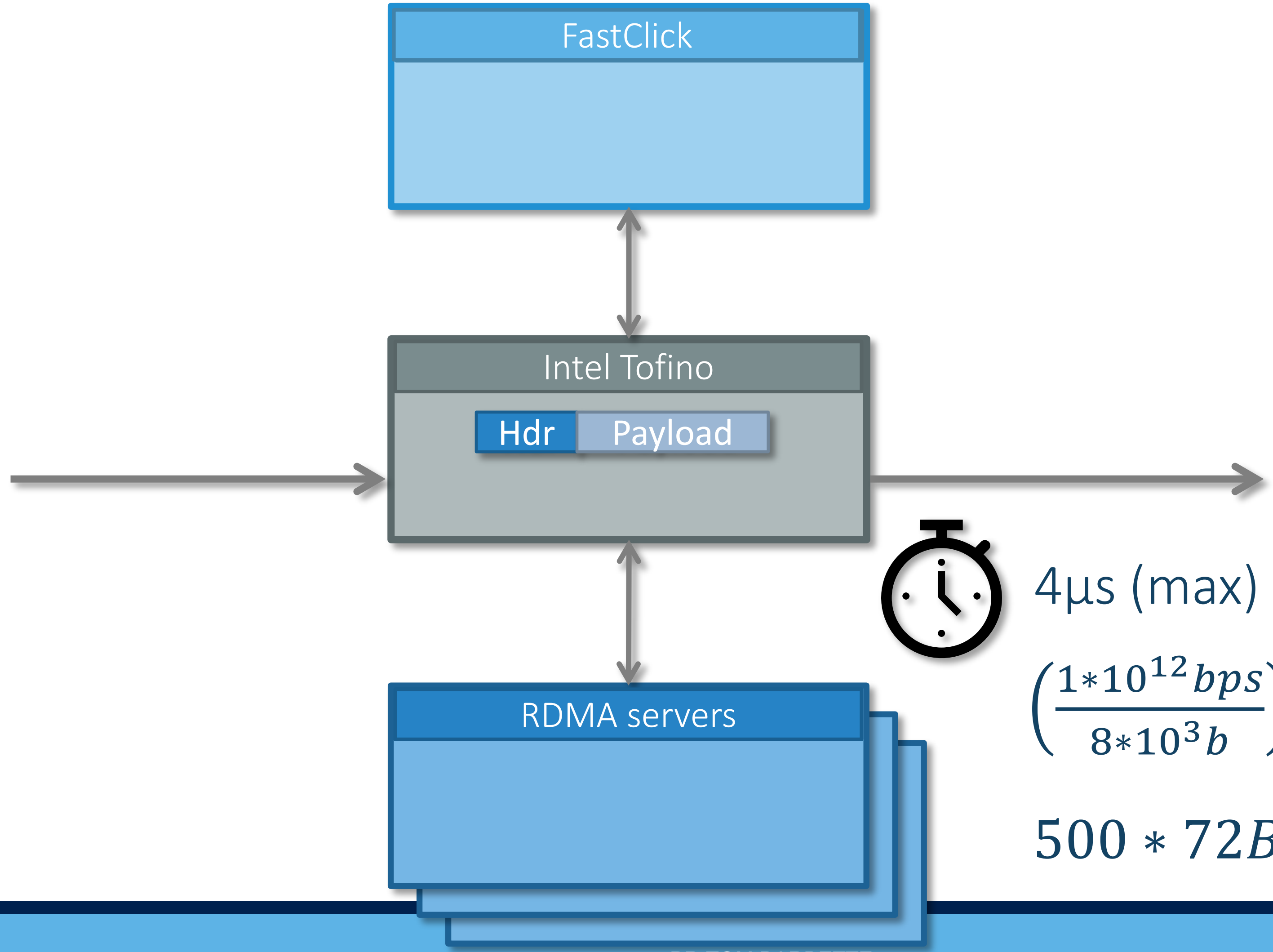


UCLouvain

Store Headers on the Switch



Store Headers on the Switch



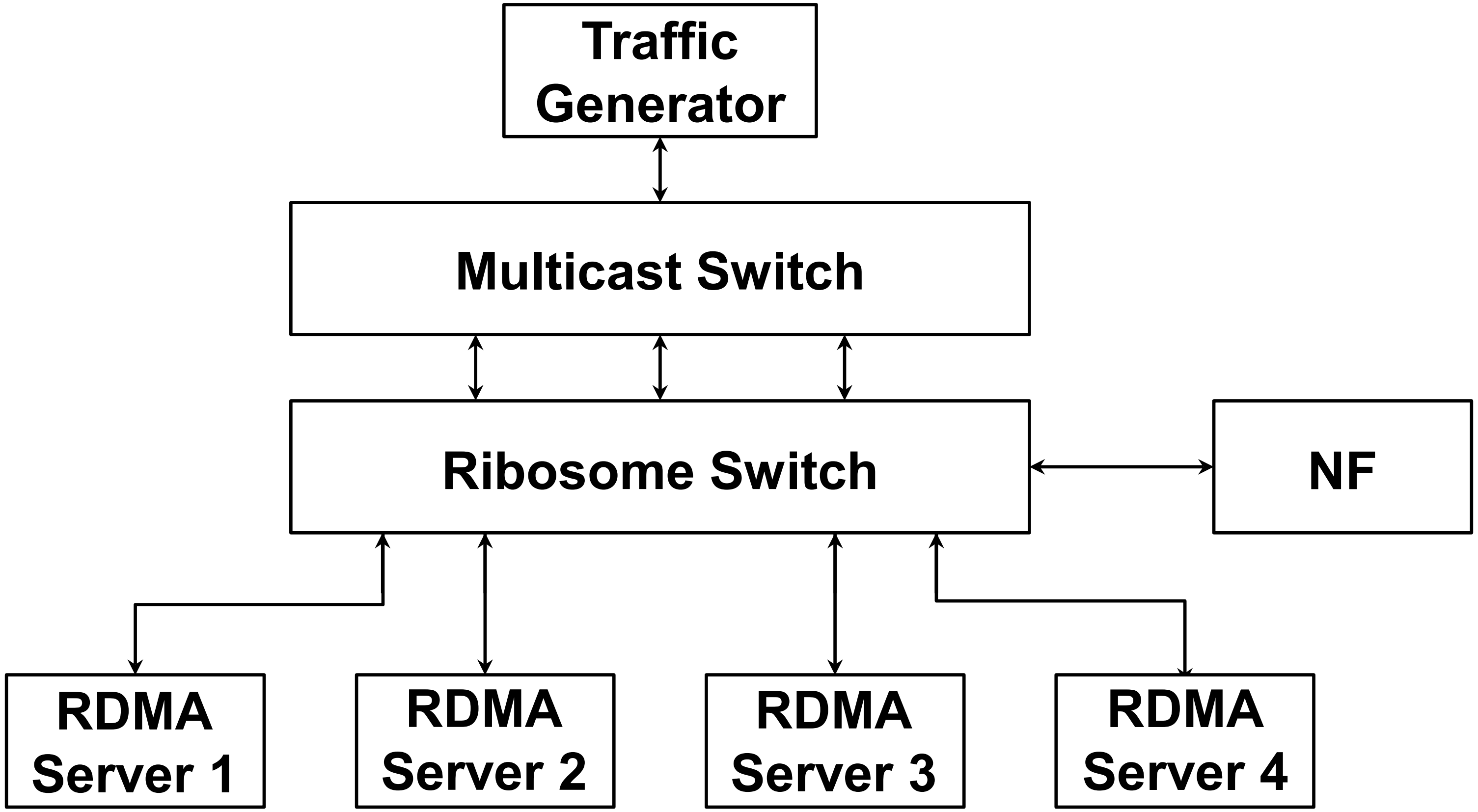
4μs (max)

$$\left(\frac{1*10^{12}bps}{8*10^{3}b}\right) * 4\mu s = 500 \text{ headers}$$

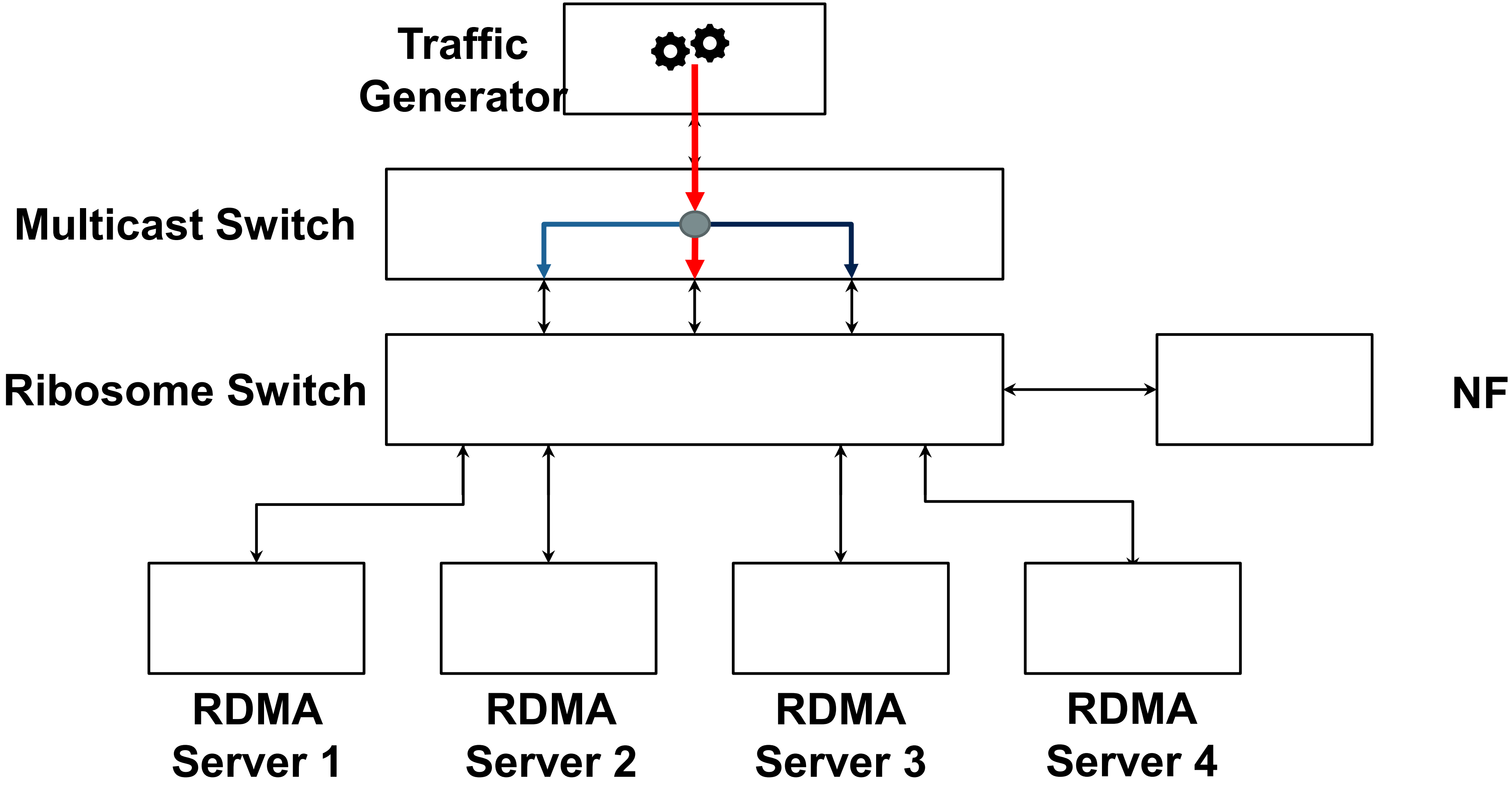
$$500 * 72B = 36000B$$ ✓

Evaluation

Testbed and Workload Generation

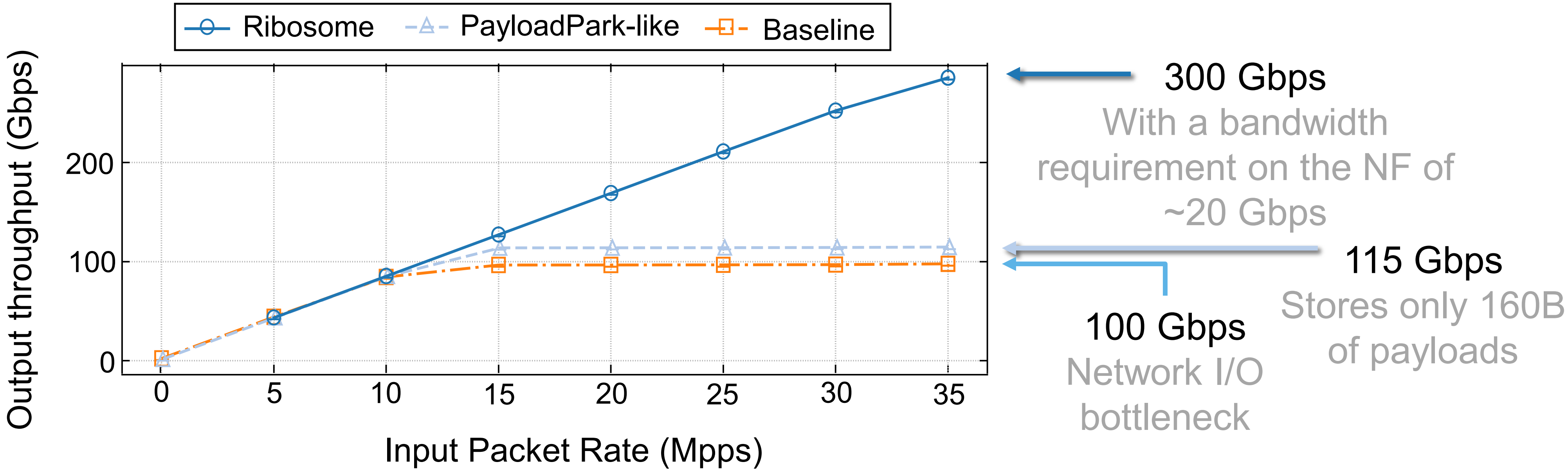


Testbed and Workload Generation



Throughput Gain

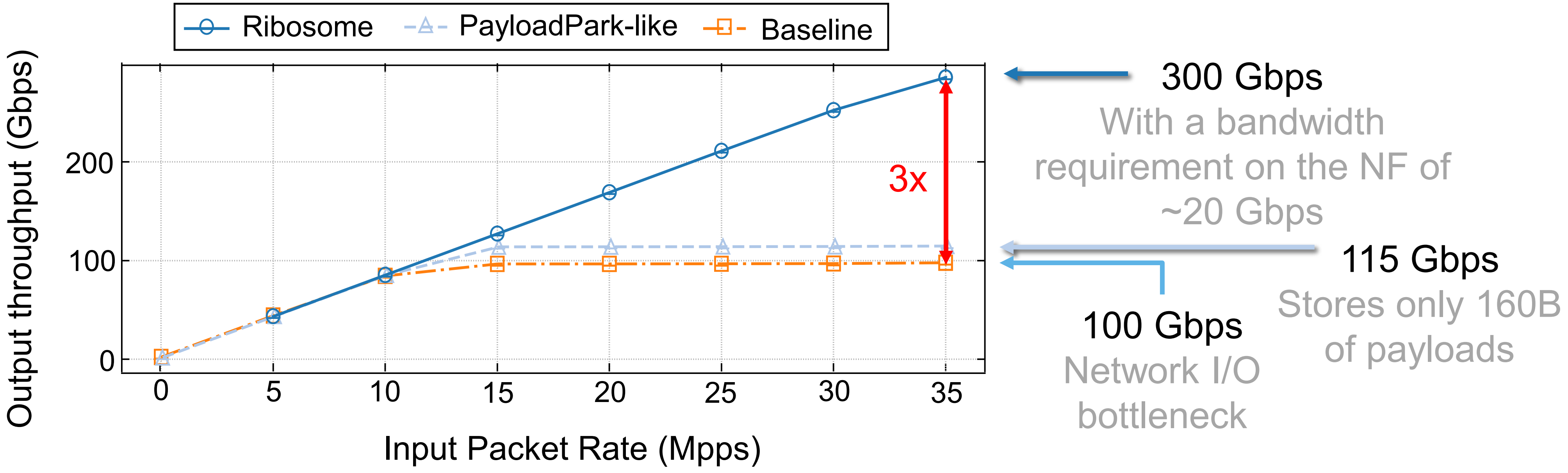
How much Ribosome improves the per-packet throughput on the NF server?



Tested NF: Forwarder

Throughput Gain

How much Ribosome improves the per-packet throughput on the NF server?

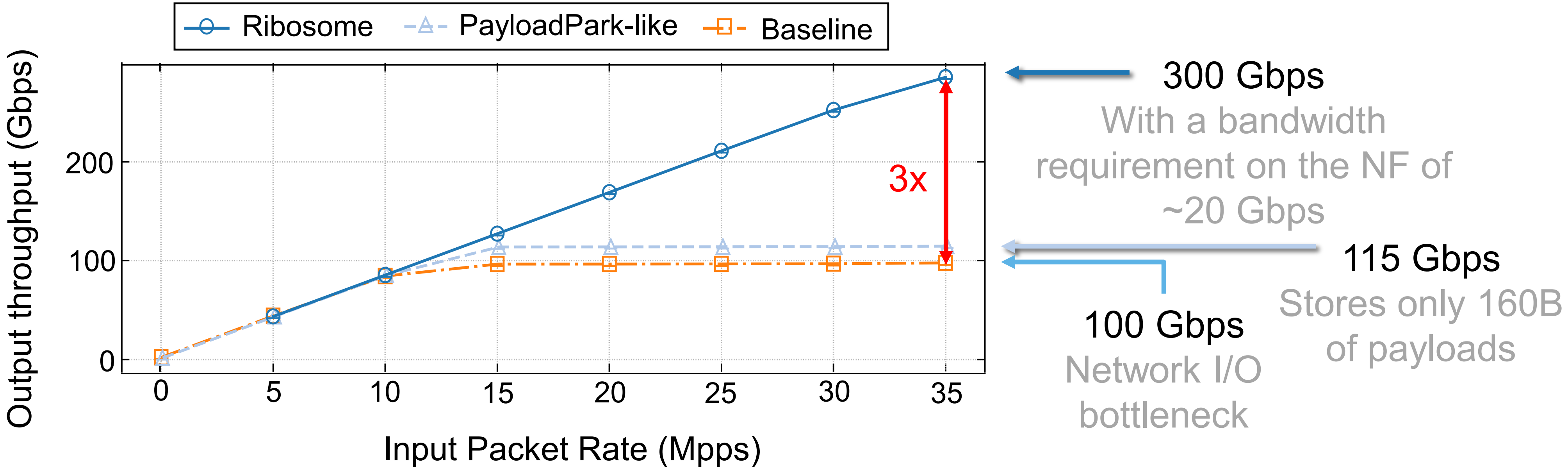


Tested NF: Forwarder

Throughput Gain

How much Ribosome improves the per-packet throughput on the NF server?

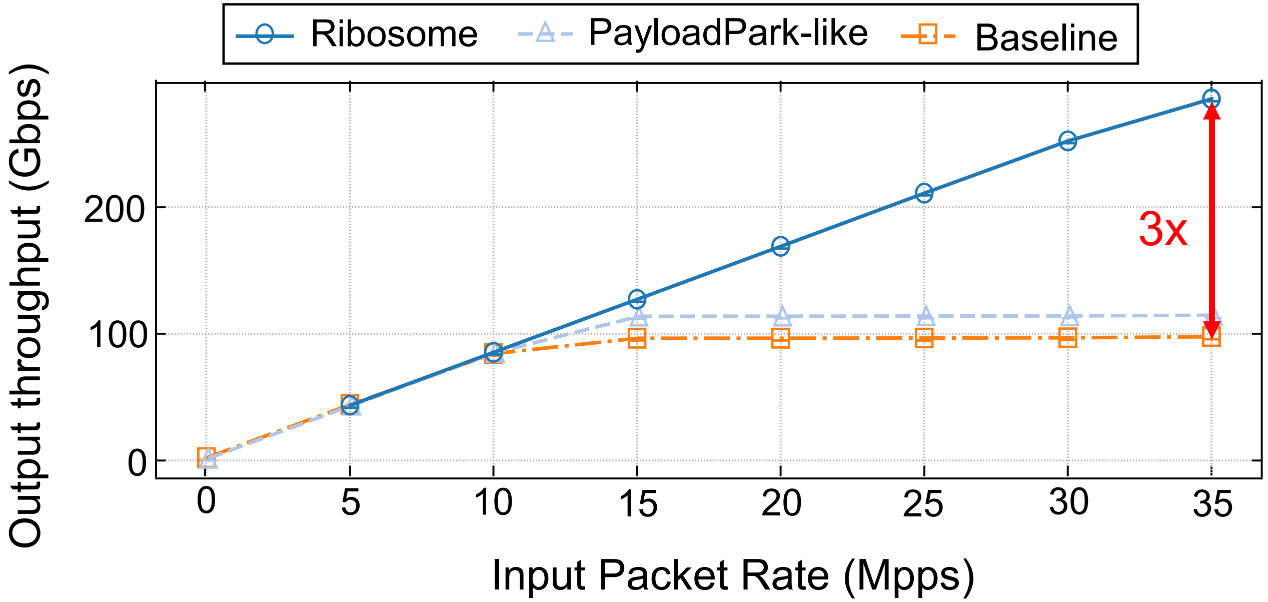
Ribosome enables multi-100Gbps packet processing!



Throughput Gain

How much Ribosome improves the per-packet throughput on the NF server?

Ribosome enables multi-100Gbps packet processing!



~75Gbps for RDMA server
Due to RDMA overheads

300 Gbps
With a bandwidth requirement on the NF of ~20 Gbps

115 Gbps
Stores only 160B of payloads

100 Gbps
Network I/O bottleneck

Throughput Gain

How much Ribosome improves the per-packet throughput on the NF server?

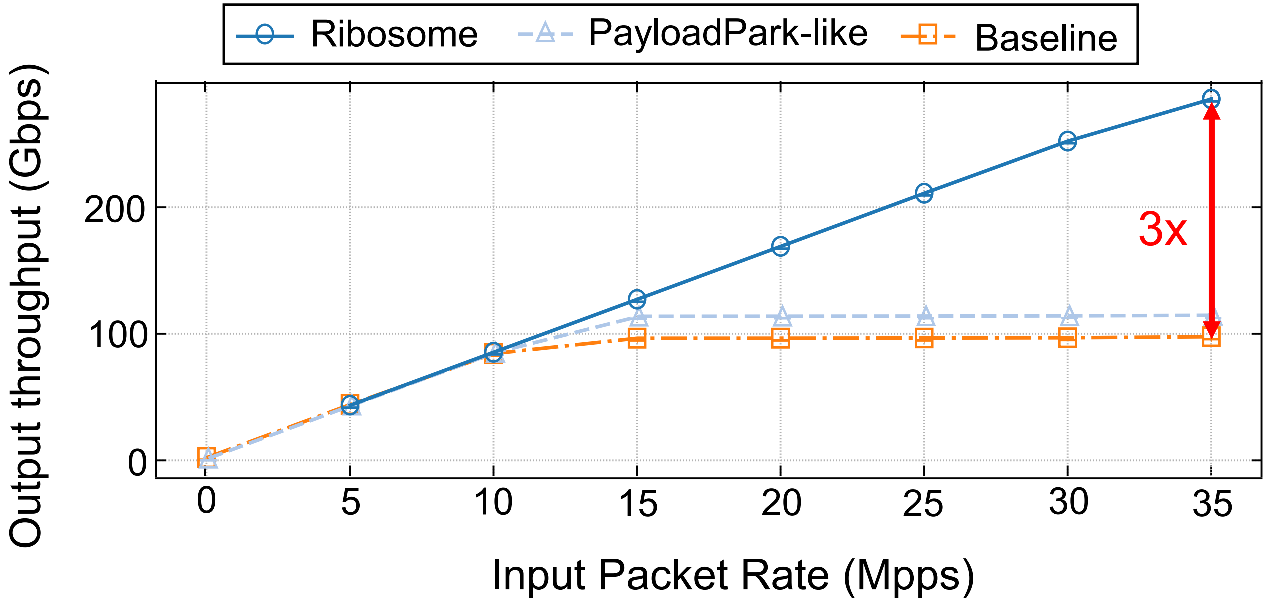
Ribosome enables multi-100Gbps packet processing!

How many servers to potentially process 1Tbps?

14 shared RDMA servers

~75Gbps for RDMA server

Due to RDMA overheads



300 Gbps

With a bandwidth requirement on the NF of ~20 Gbps

115 Gbps

Stores only 160B of payloads

100 Gbps

Network I/O bottleneck

Throughput Gain

How much Ribosome improves the per-packet throughput on the NF server?

Ribosome enables multi-100Gbps packet processing!

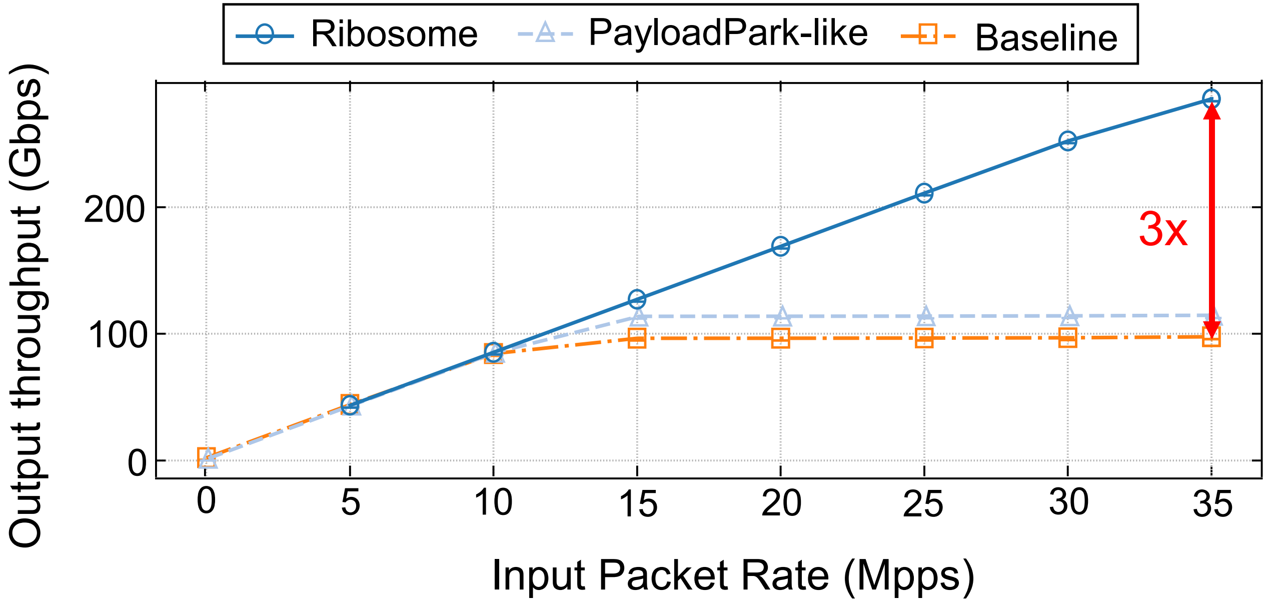
A datacenter has thousands of servers!

How many servers to potentially process 1Tbps?

14 shared RDMA servers

~75Gbps for RDMA server

Due to RDMA overheads



300 Gbps

With a bandwidth requirement on the NF of ~20 Gbps

115 Gbps

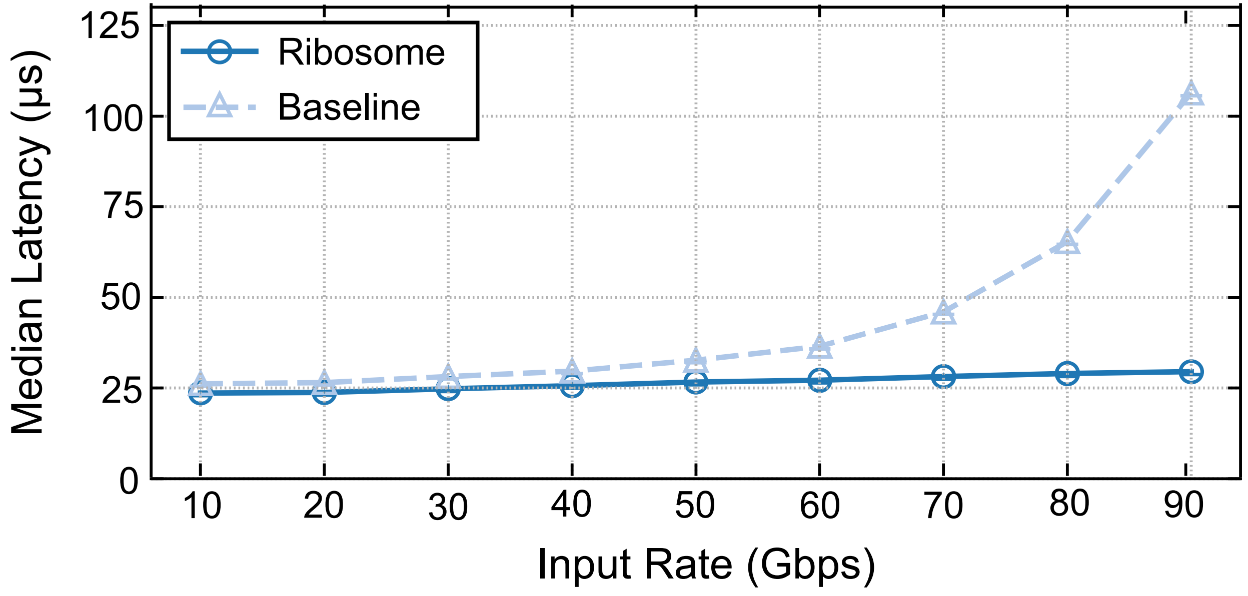
Stores only 160B of payloads

100 Gbps

Network I/O bottleneck

Latency Gain

How much Ribosome improves the latency gain on the NF server?



~ 110 μs

Due to **congestioned queues** and the **high transmission rate**

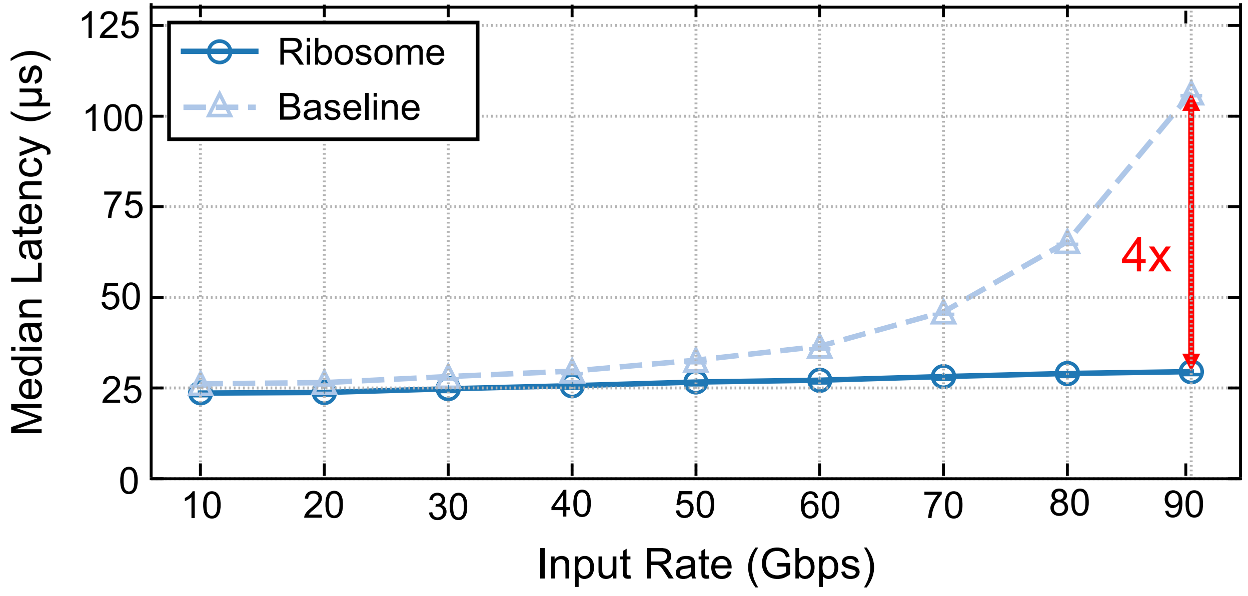
~25 μs

Almost **constant**

Tested NF: Forwarder

Latency Gain

How much Ribosome improves the latency gain on the NF server?



~ 110 µs

Due to **congestioned queues** and the **high trasmission rate**

~25 µs

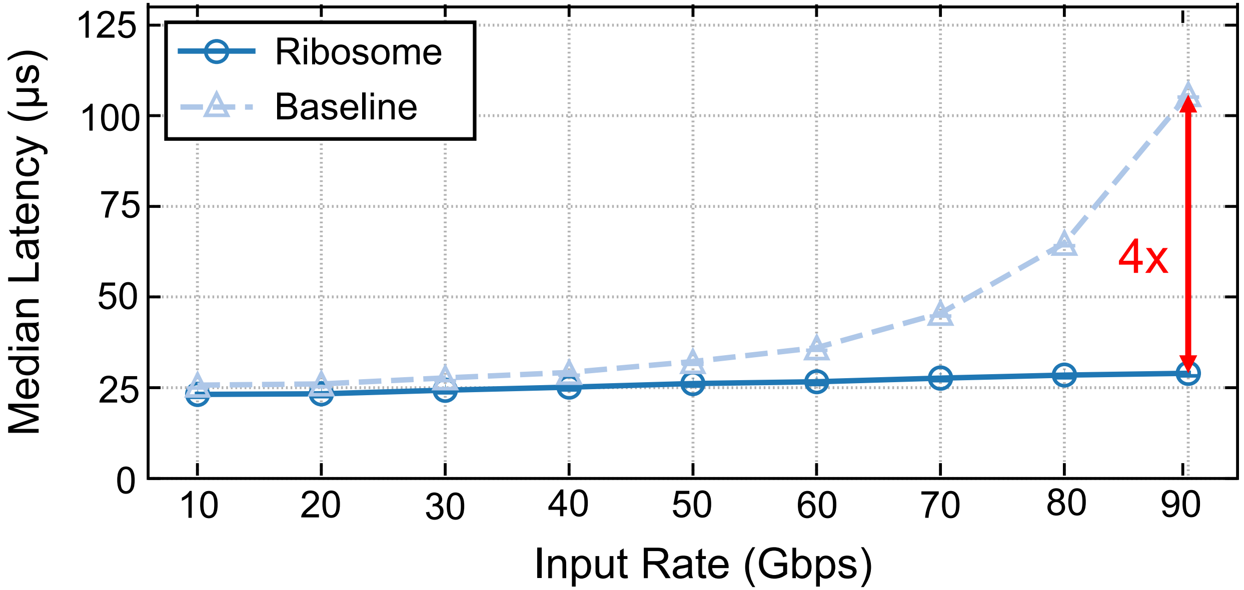
Almost **constant**

Tested NF: Forwarder

Latency Gain

How much Ribosome improves the latency gain on the NF server?

And the tail latency? → Similar trend!



~ 110 µs

Due to **congestioned queues** and the **high trasmission rate**

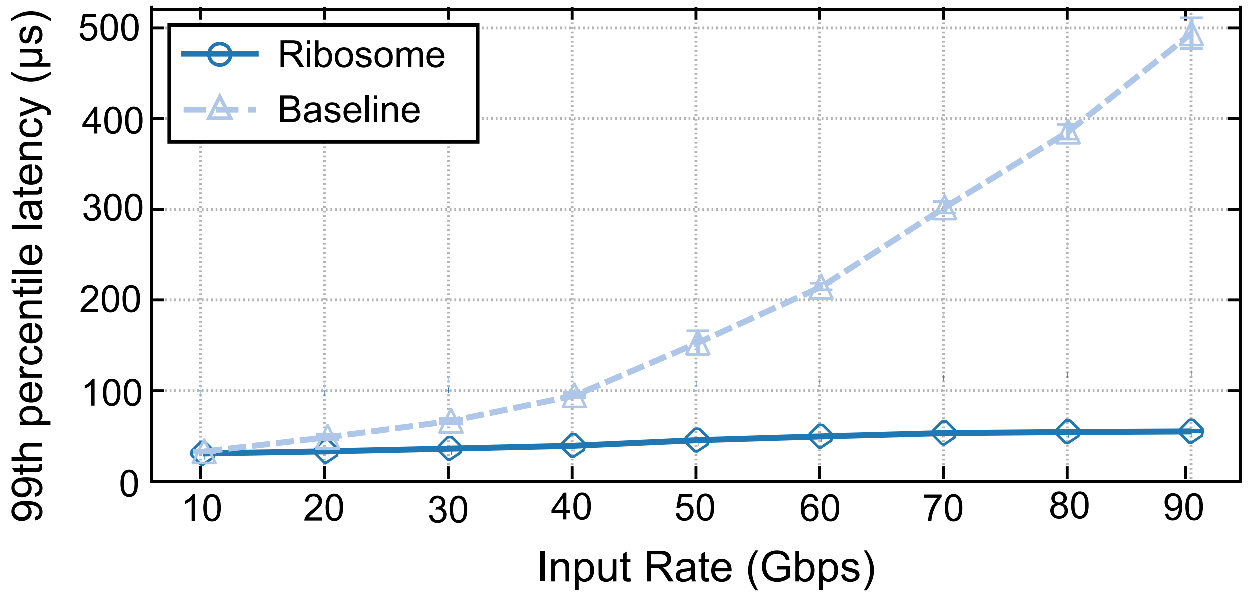
~25 µs

Almost **constant**

Latency Gain

How much Ribosome improves the latency gain on the NF server?

And the tail latency? → Similar trend!



~ 500 µs

Due to **congestioned queues** and the **high trasmission rate**

~60 µs

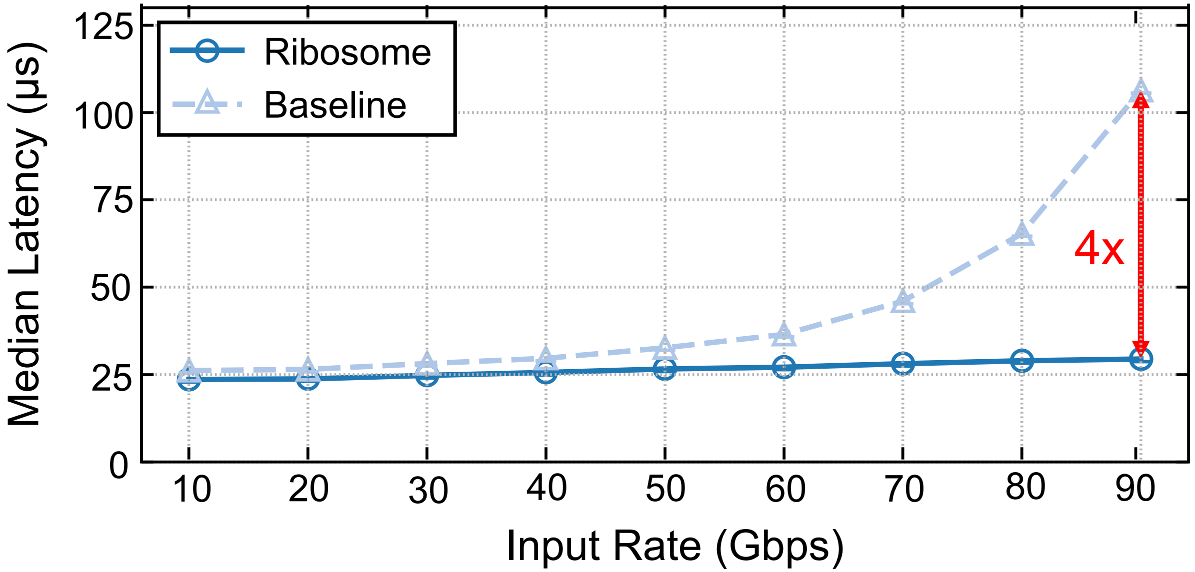
Almost **constant**

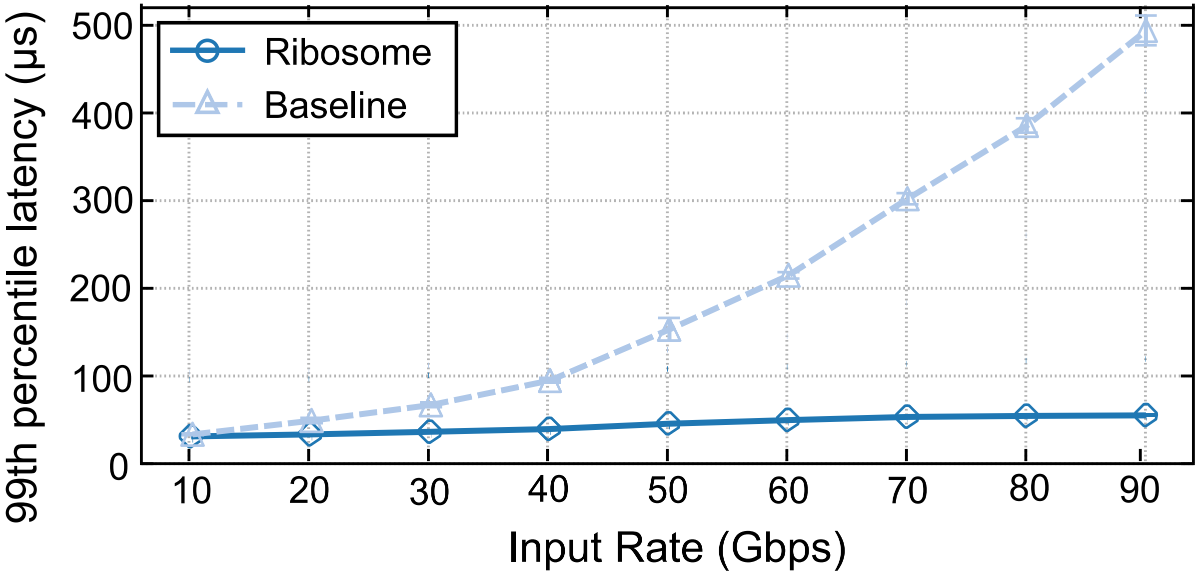
Latency Gain

How much Ribosome improves the latency gain on the NF server?

Reducing queue sizes and the input throughput on the NF reduce latency!

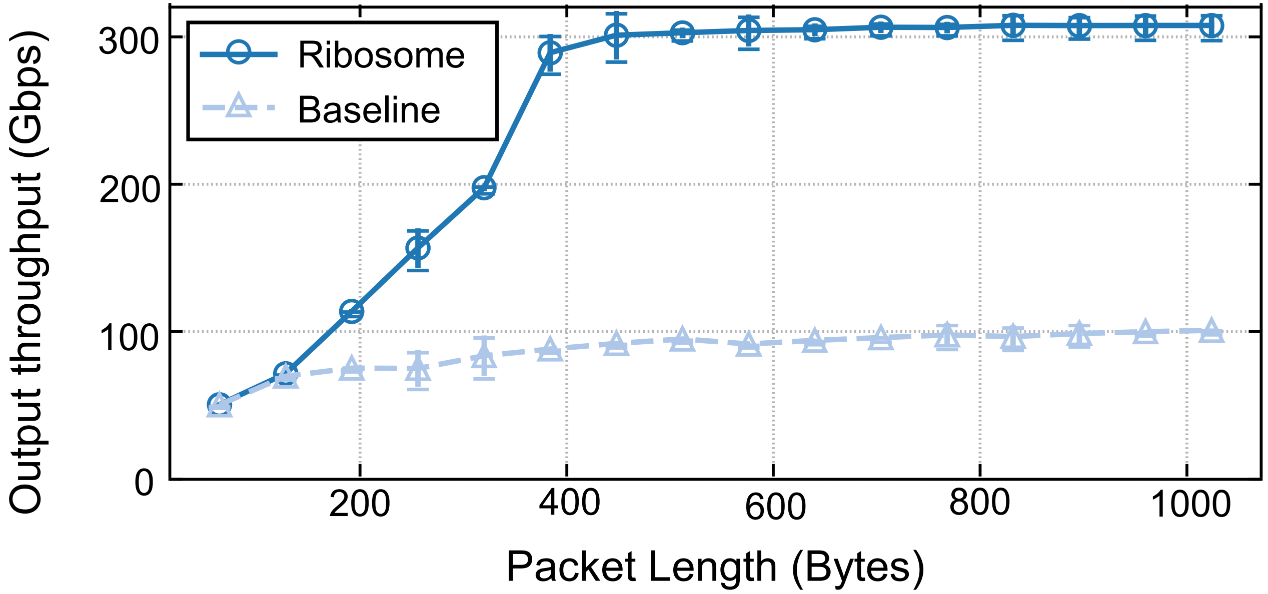
And the tail latency? → Similar trend!





Packet Size Impact

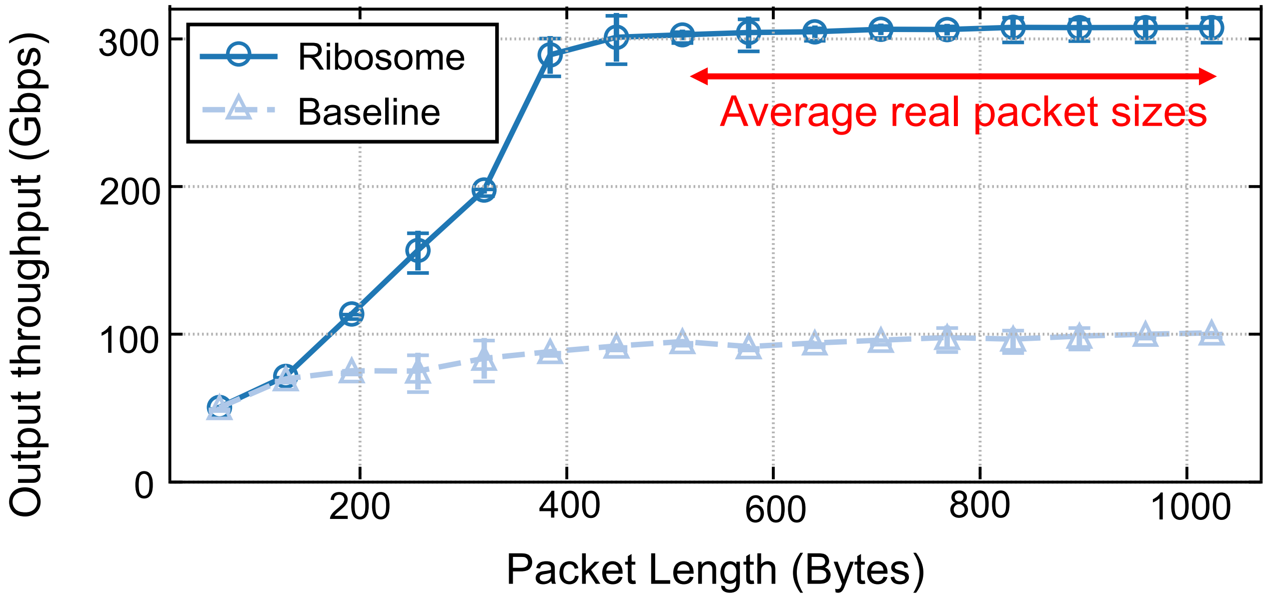
How does the packet size impact the throughput gains?



Tested NF: Forwarder

Packet Size Impact

How does the packet size impact the throughput gains?

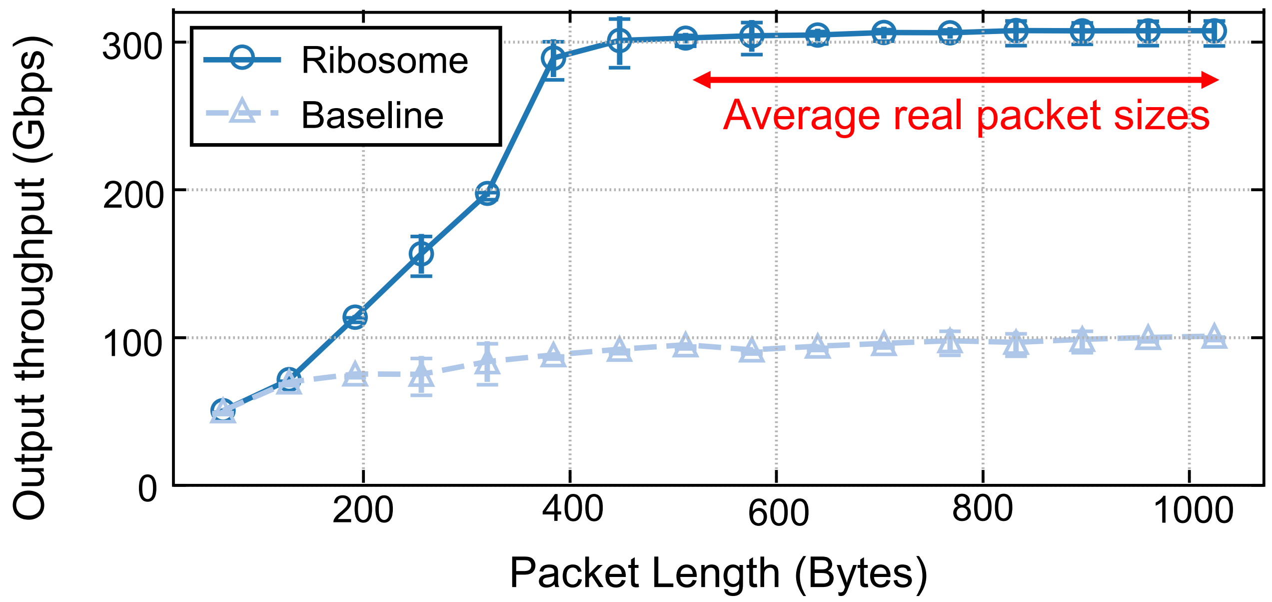


Tested NF: Forwarder

Packet Size Impact

How does the packet size impact the throughput gains?

Highly effective for relevant real-world scenarios!

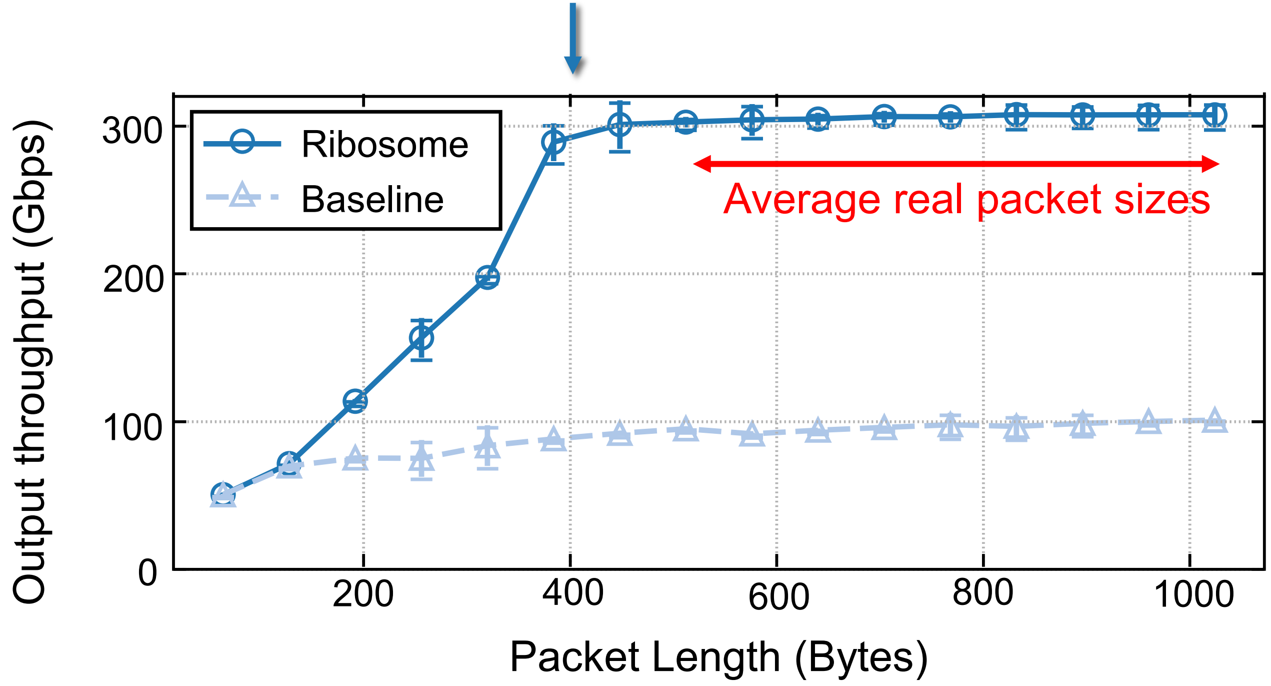


Packet Size Impact

How does the packet size impact the throughput gains?

Highly effective for relevant real-world scenarios!

300Gbps
With 400B packets



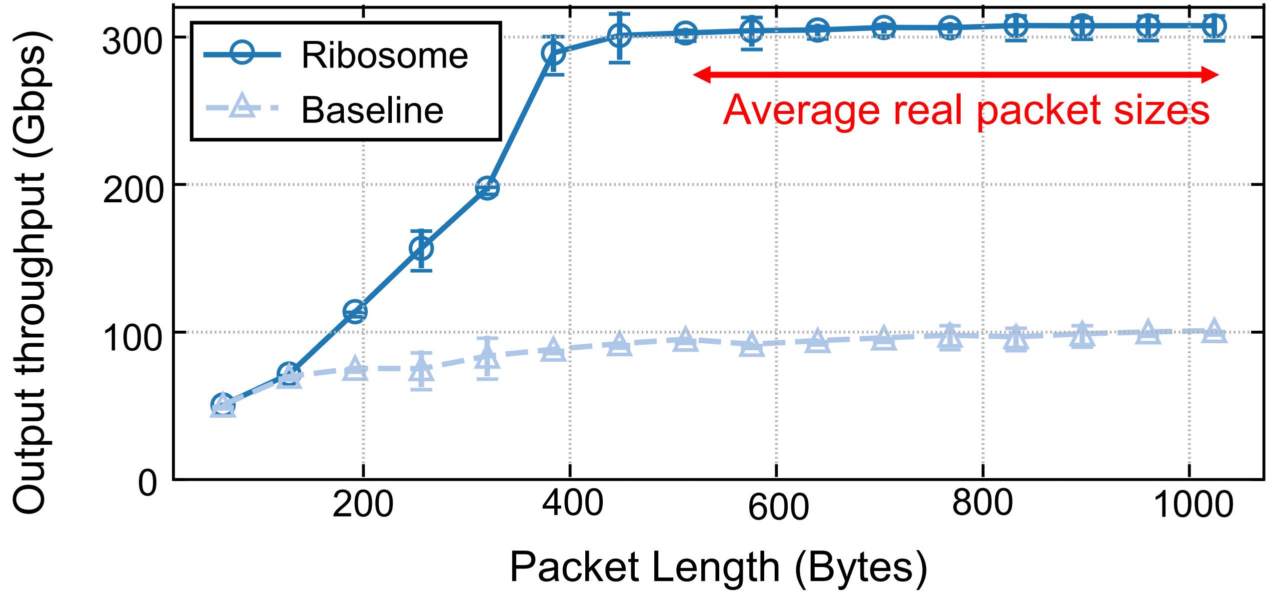
Packet Size Impact

How does the packet size impact the throughput gains?

Highly effective for relevant real-world scenarios!

300Gbps
With 400B packets

Corresponds to ~93Mpps
NF can handle them!



Packet Size Impact

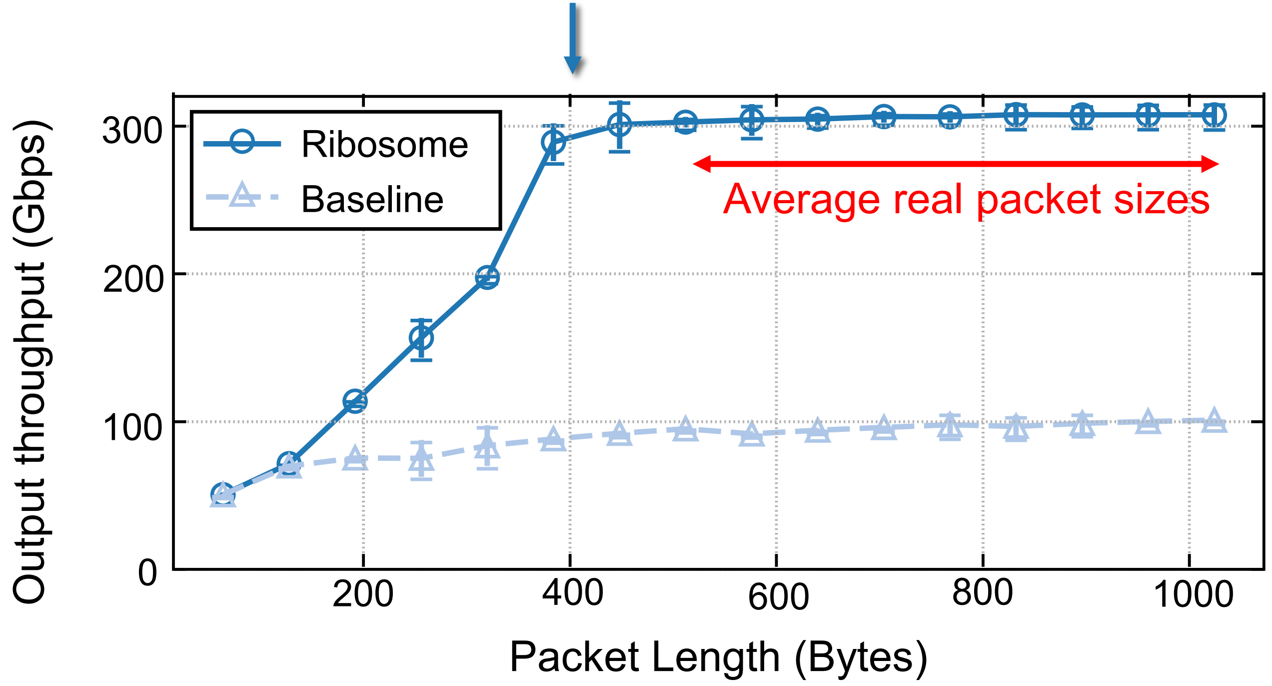
How does the packet size impact the throughput gains?

Highly effective for relevant real-world scenarios!

300Gbps
With 400B packets

Corresponds to ~93Mpps
NF can handle them!

With 93M packets of 1.5KB
We process 1Tbps!



Packet Size Impact

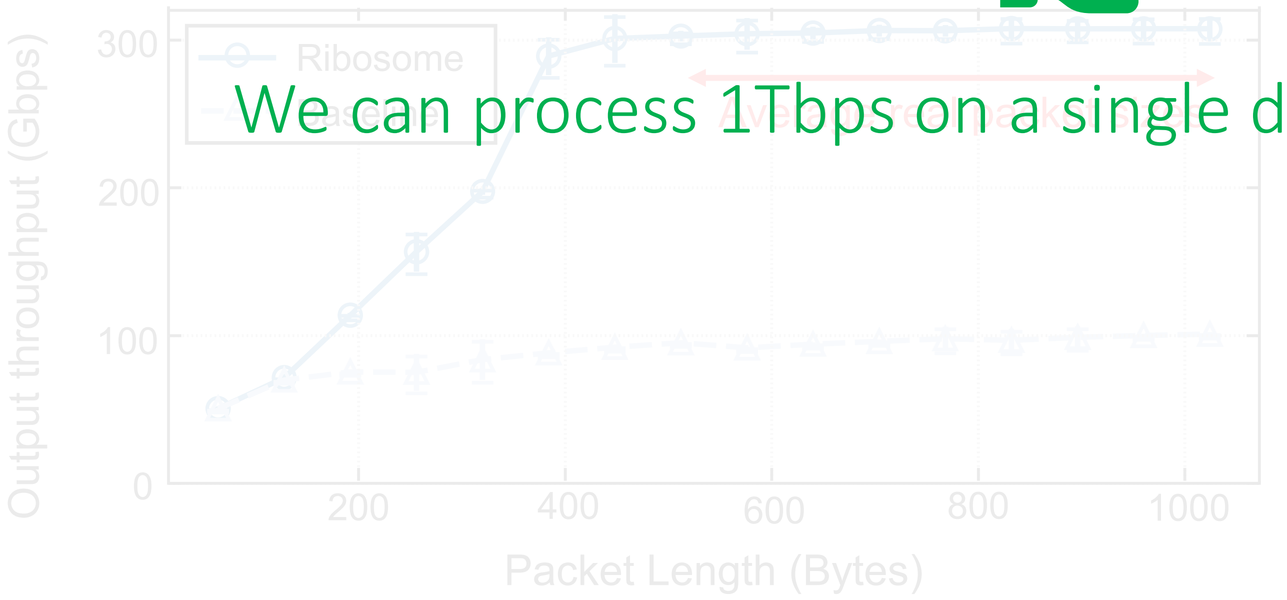
How does the packet size impact the throughput gains?

Highly effective for relevant real-world scenarios!

300Gbps
With 400B packets

Corresponds to ~93Mpps
NF can handle them!

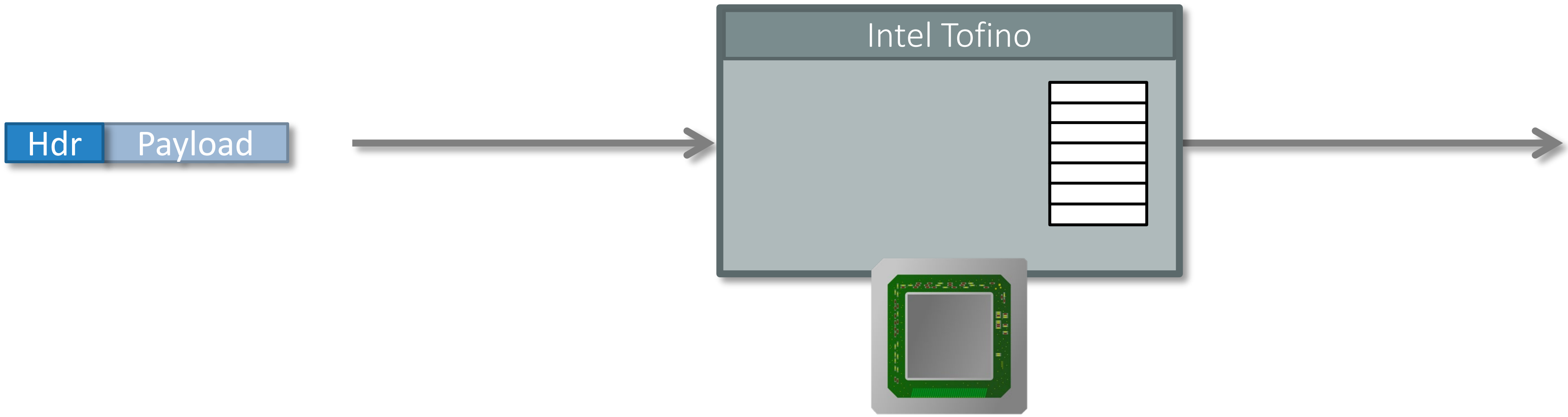
With 93M packets of 1.5KB
We process 1Tbps!



We can process 1Tbps on a single dedicated CPU!

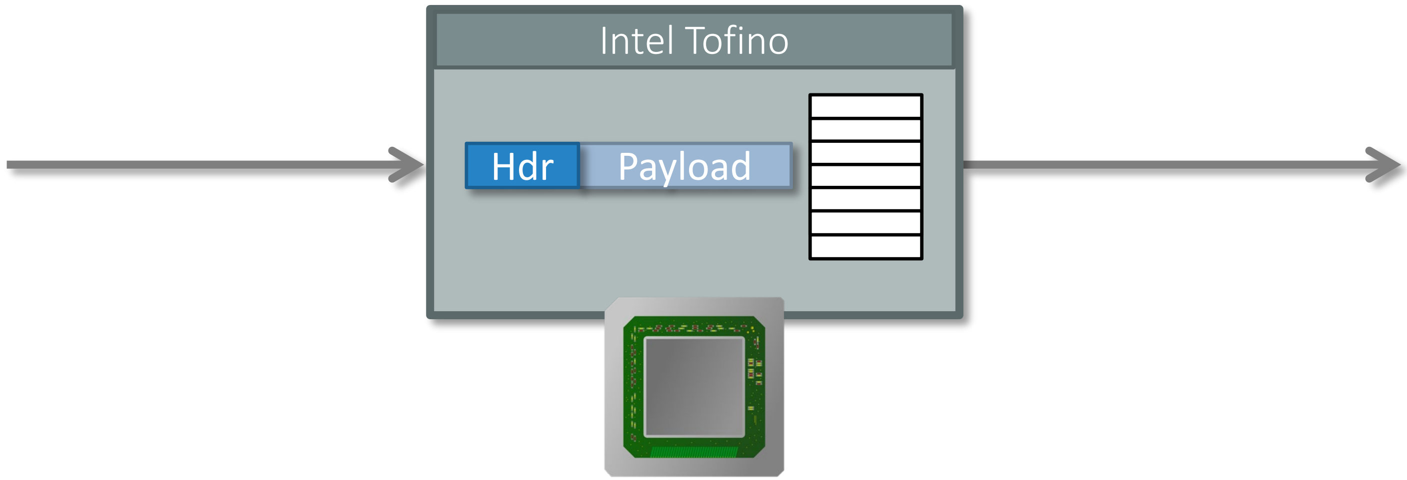
Side note : can’t switches handle connection tracking?

After all, that’s the promise of OpenFlow.



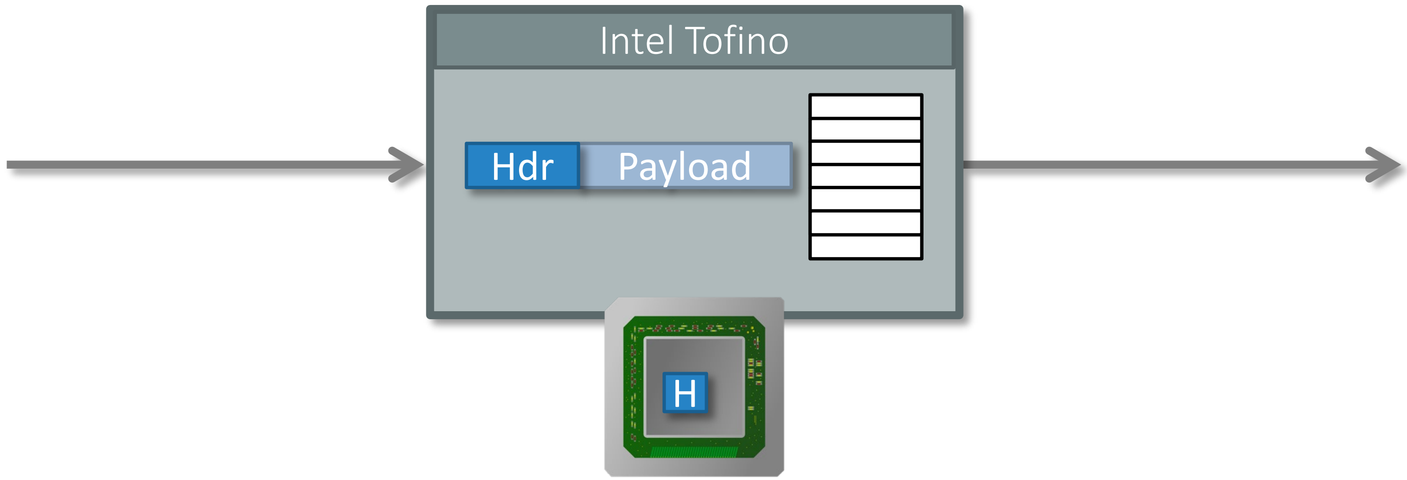
Side note : can't switches handle connection tracking?

After all, that's the promise of OpenFlow.



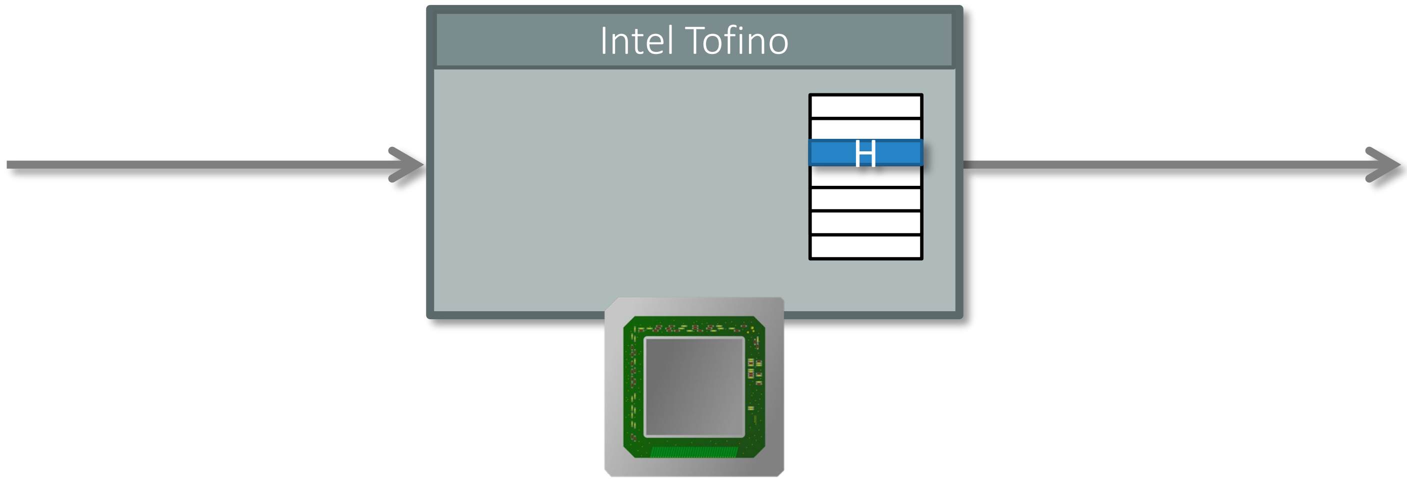
Side note : can't switches handle connection tracking?

After all, that's the promise of OpenFlow.



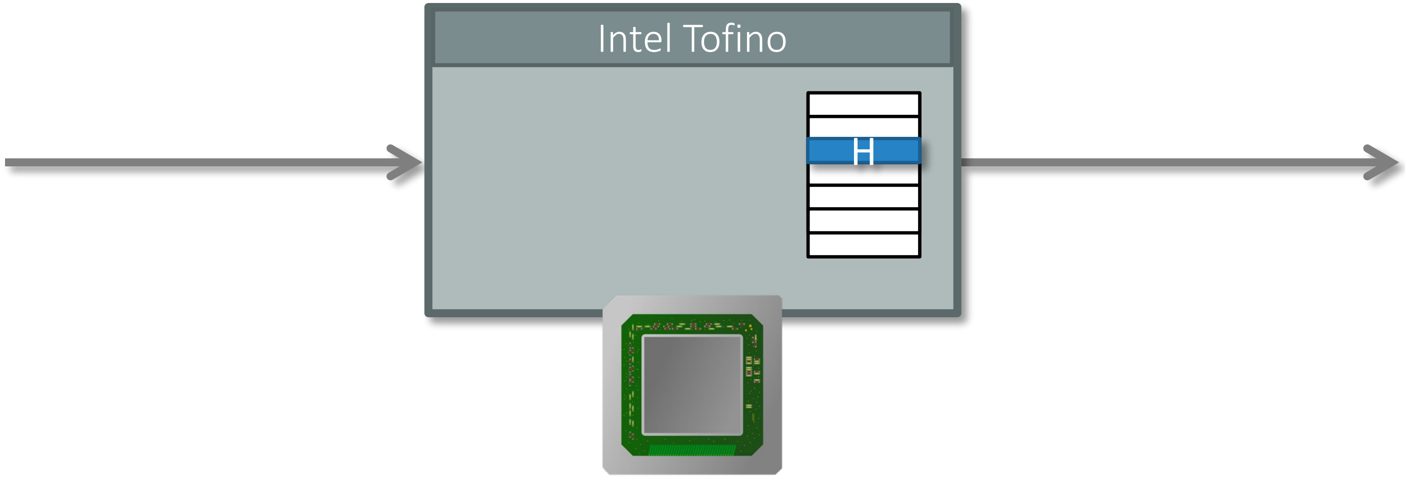
Side note : can't switches handle connection tracking?

After all, that's the promise of OpenFlow.



Side note : can't switches handle connection tracking?

After all, that's the promise of OpenFlow.



NO

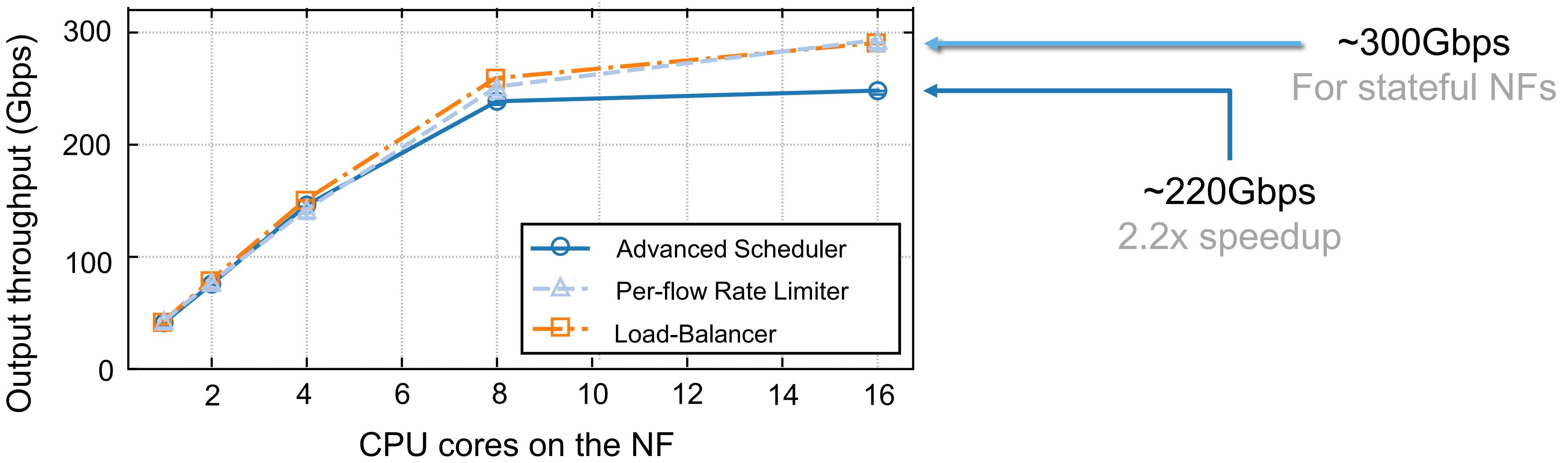
TEA [SIGCOMM'20] → Tofino has a maximum of 100k flows/seconds

CPU → Around 7M/core

OpenFlow switches are around 40K at best

Advanced Network Functions

Can we build advanced NFs on top of Ribosome?



Advanced Scheduler → Reframer [NSDI'22]

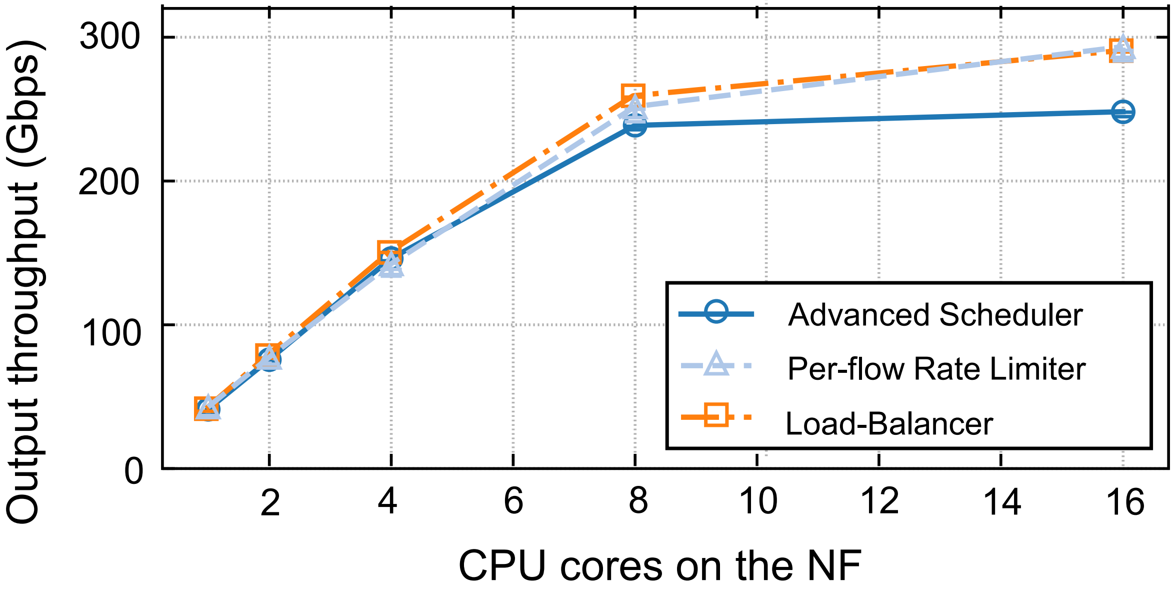
Advanced Network Functions

Can we build advanced NFs on top of Ribosome?

Ribosome supports advanced NFs!

Ribosome moves the NF bottleneck on the CPU!

→ Back to software !

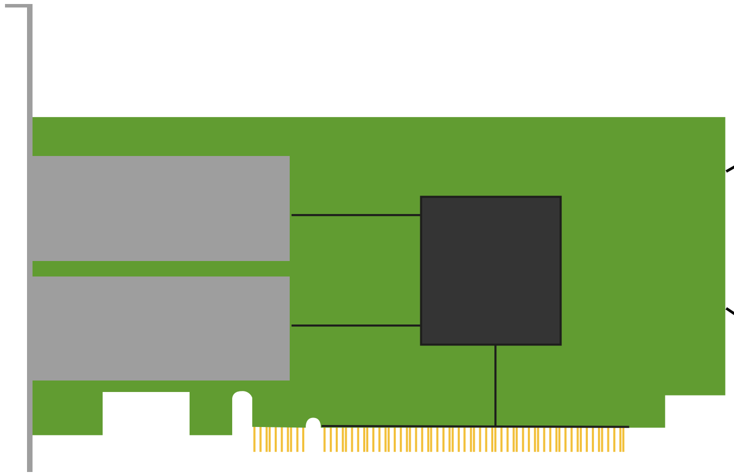


~300Gbps
For stateful NFs

~220Gbps
2.2x speedup

Can we improve the performance of connection tracking on the NF server further?

Offloading classification: what are the limits?

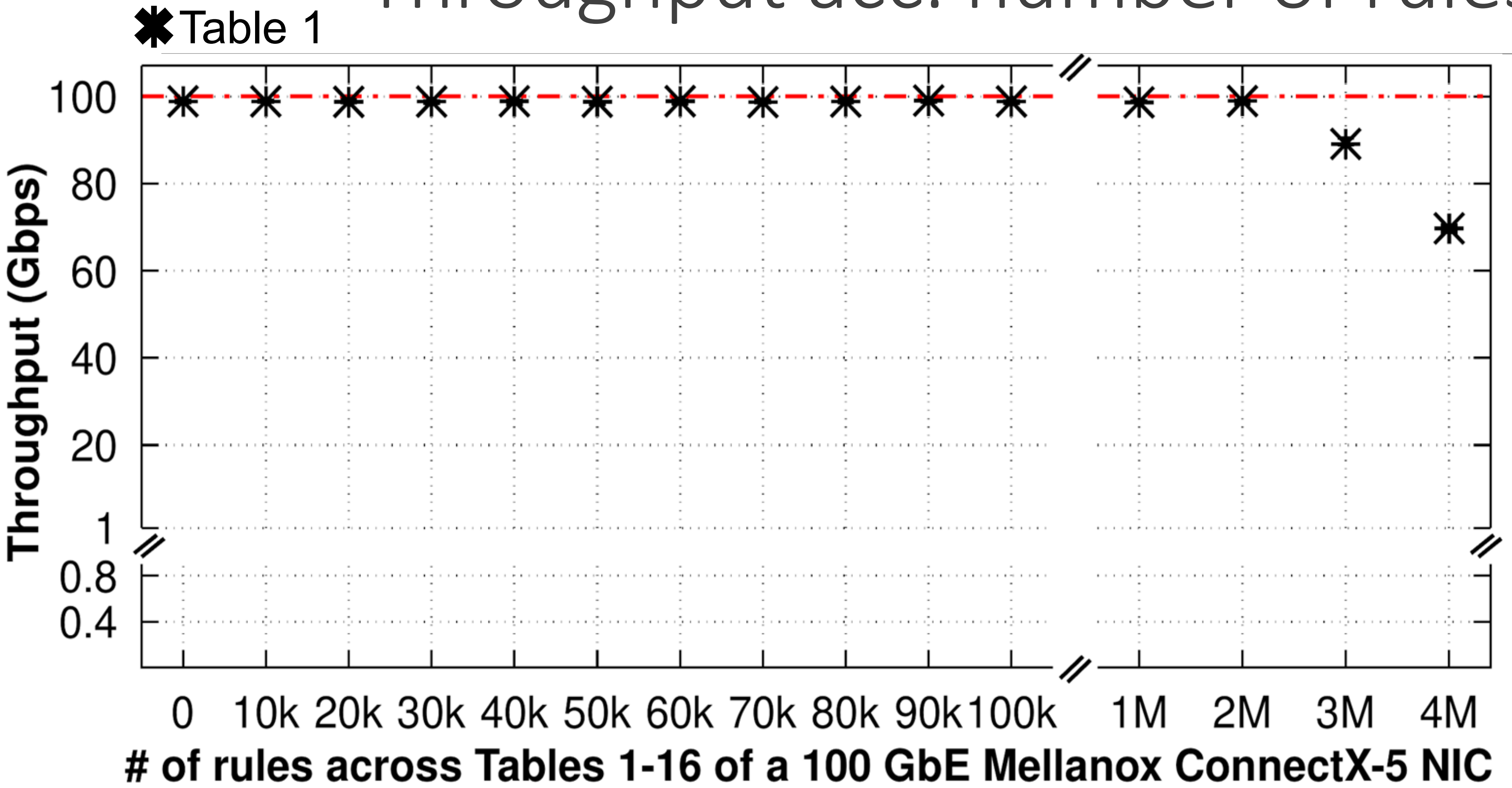


IP Source	IP Destinatio n	Action
1.1.1.1	2.2.2.2	DROP
3.3.3.3	4.4.4.4	QUEUE=0
5.5.5.5	6.6.0.0/16	RSS

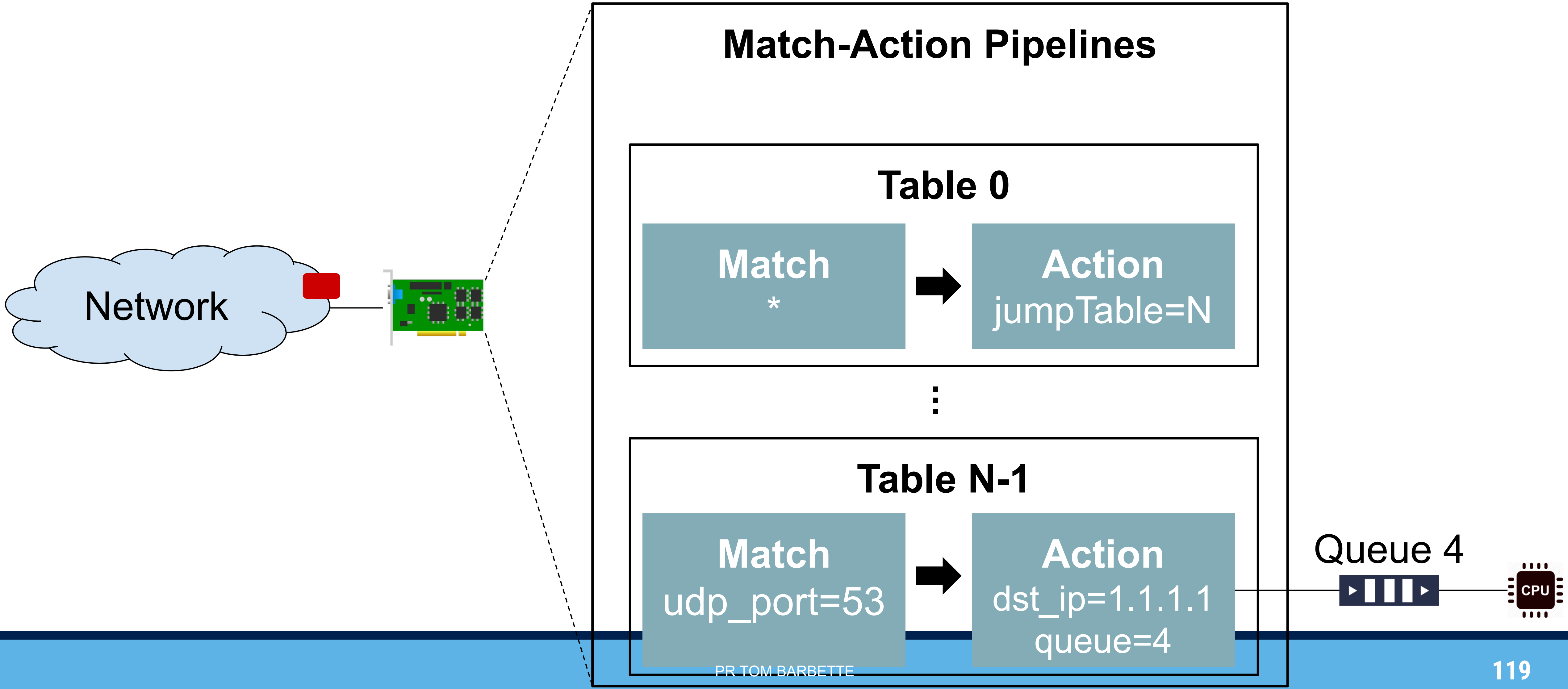
NIC-Bench
PAM’21

What you need to know about (Smart) Network Interface Cards
Open source code and results with NVIDIA ConnectX-4, ConnectX-5, ConnectX-6, and Bluefield NICs (here)
Georgios P. Katsikas, Tom Barbette, Marco Chiesa, Dejan Kostić, and Gerald Q. Maguire Jr.

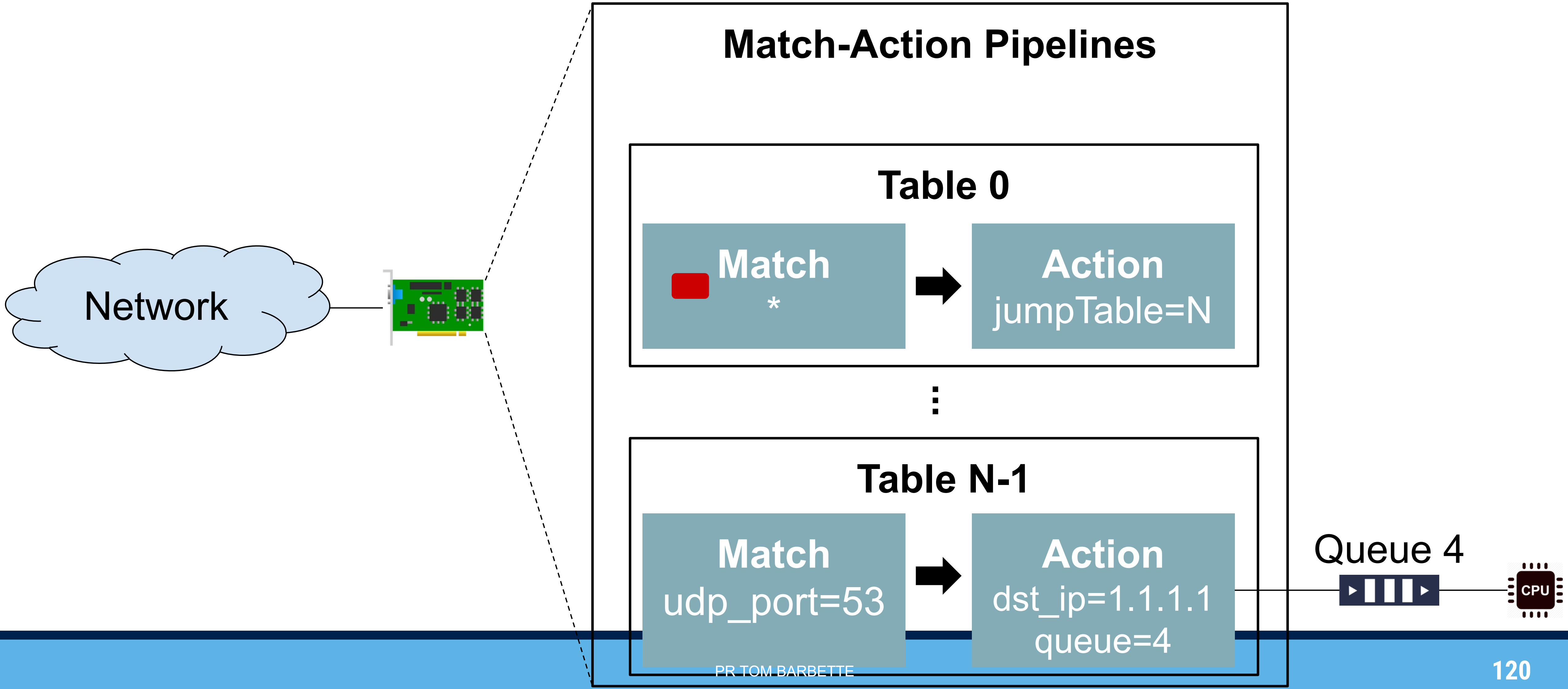
Throughput acc. number of rules



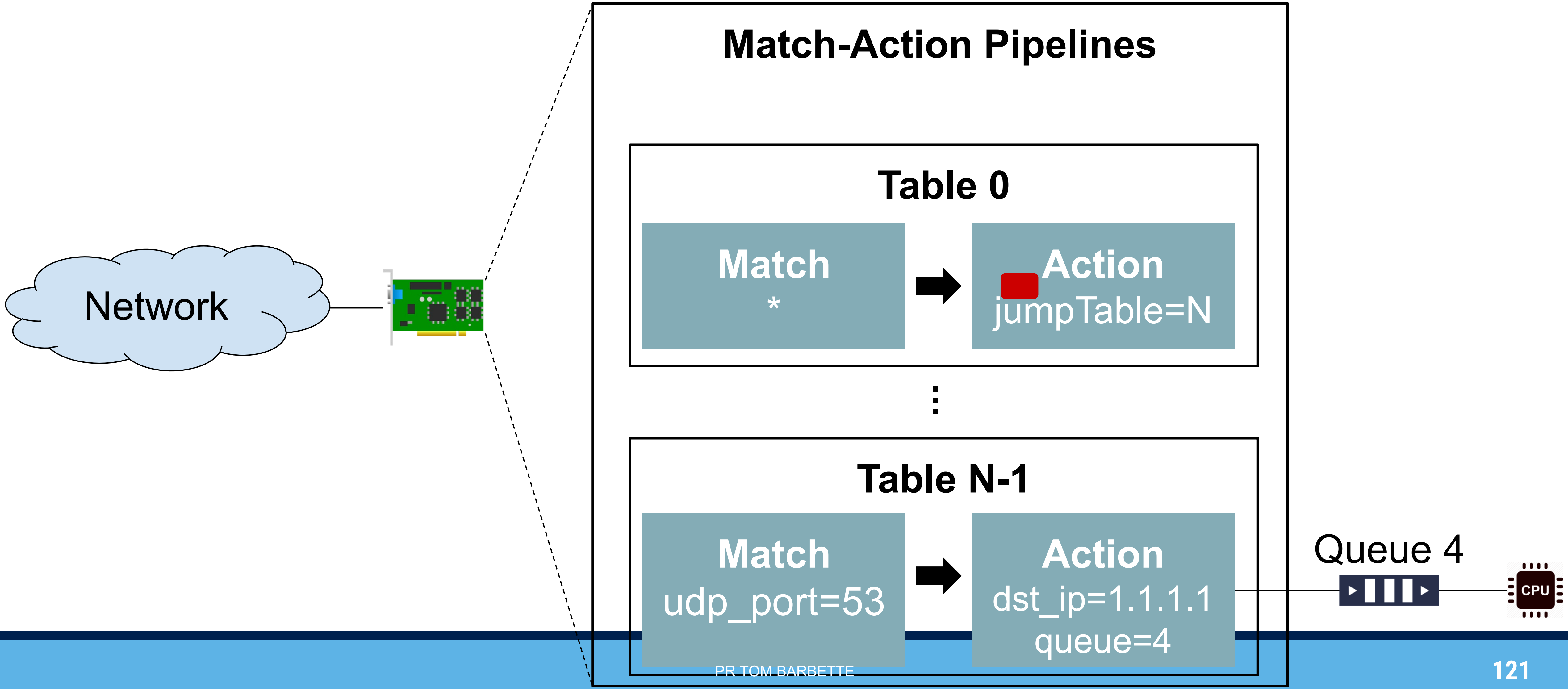
How do NICs perform processing?



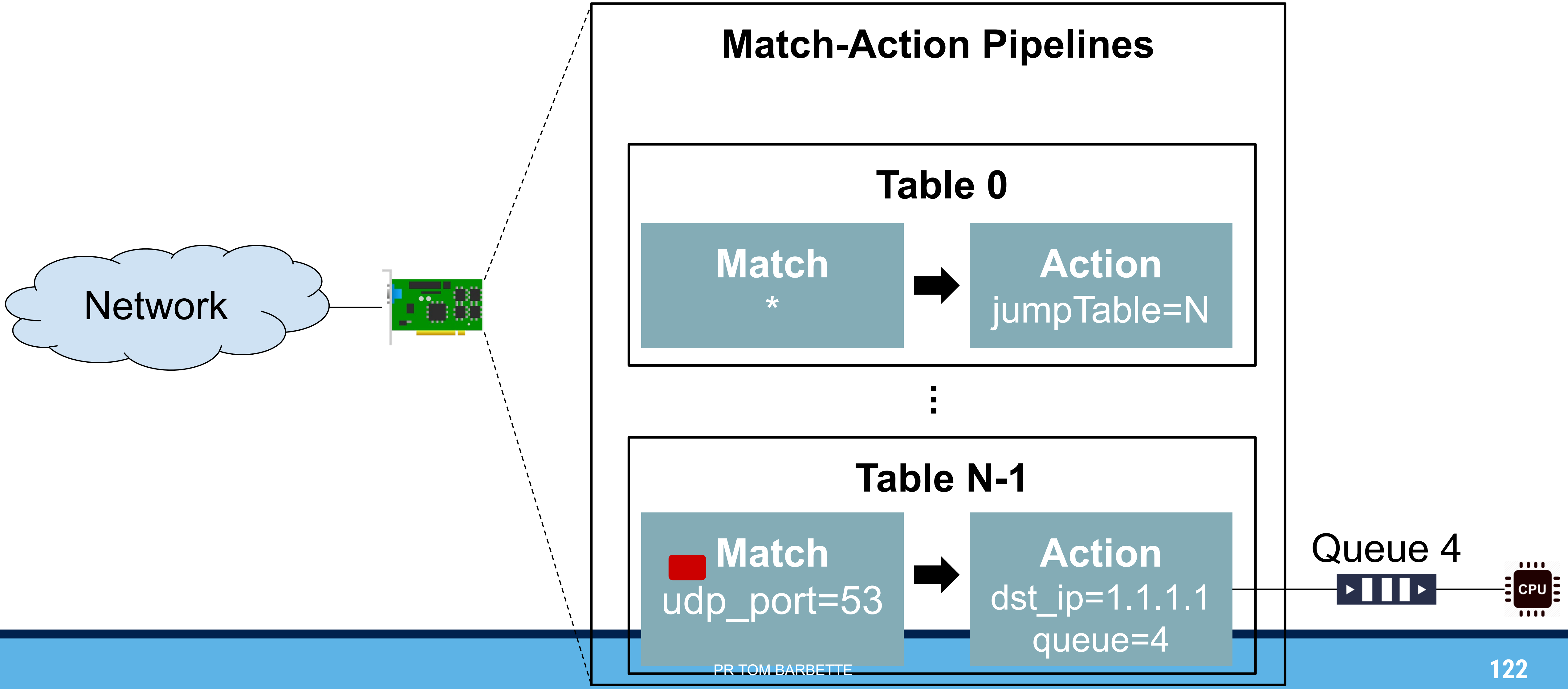
How do NICs perform processing?



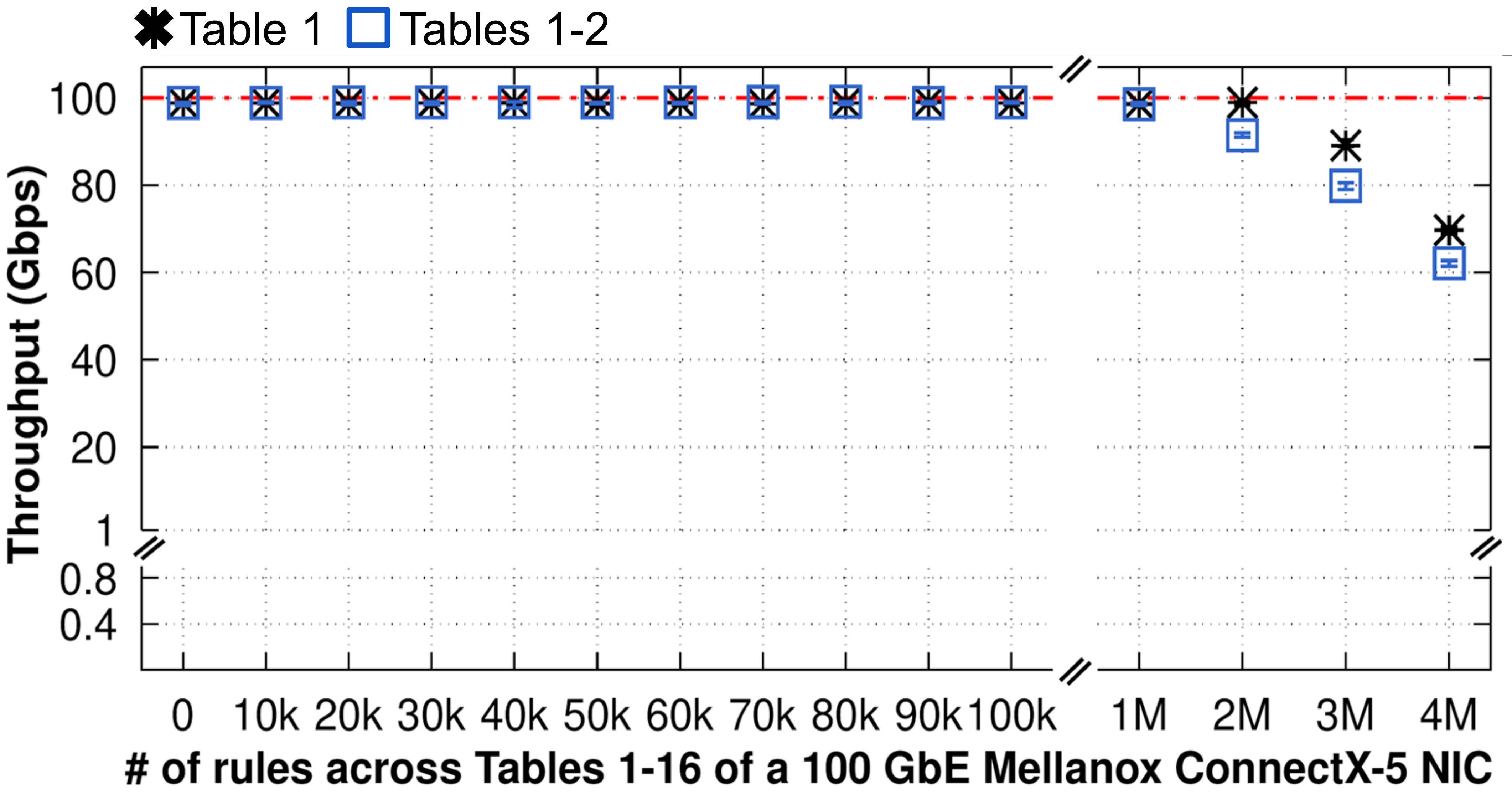
How do NICs perform processing?



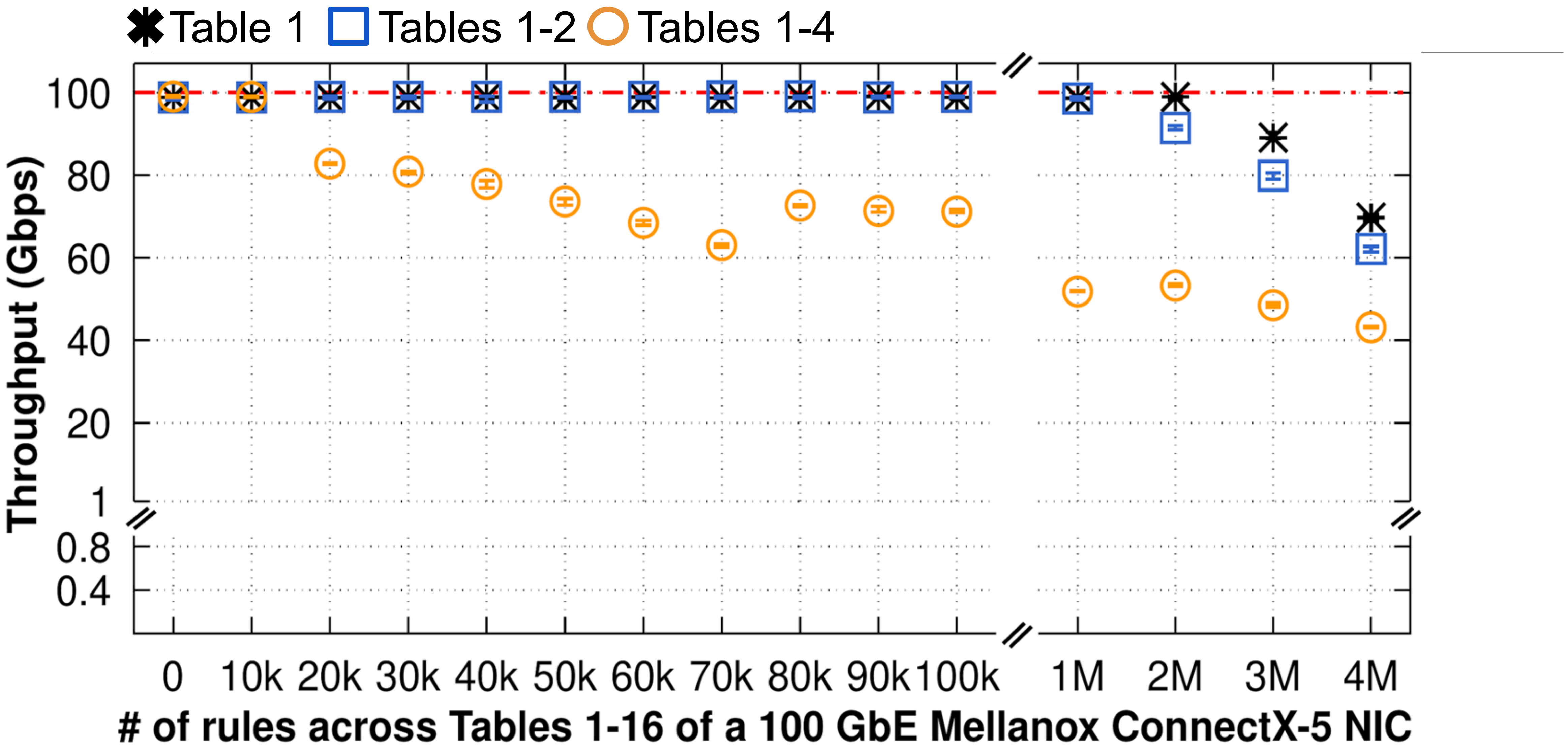
How do NICs perform processing?



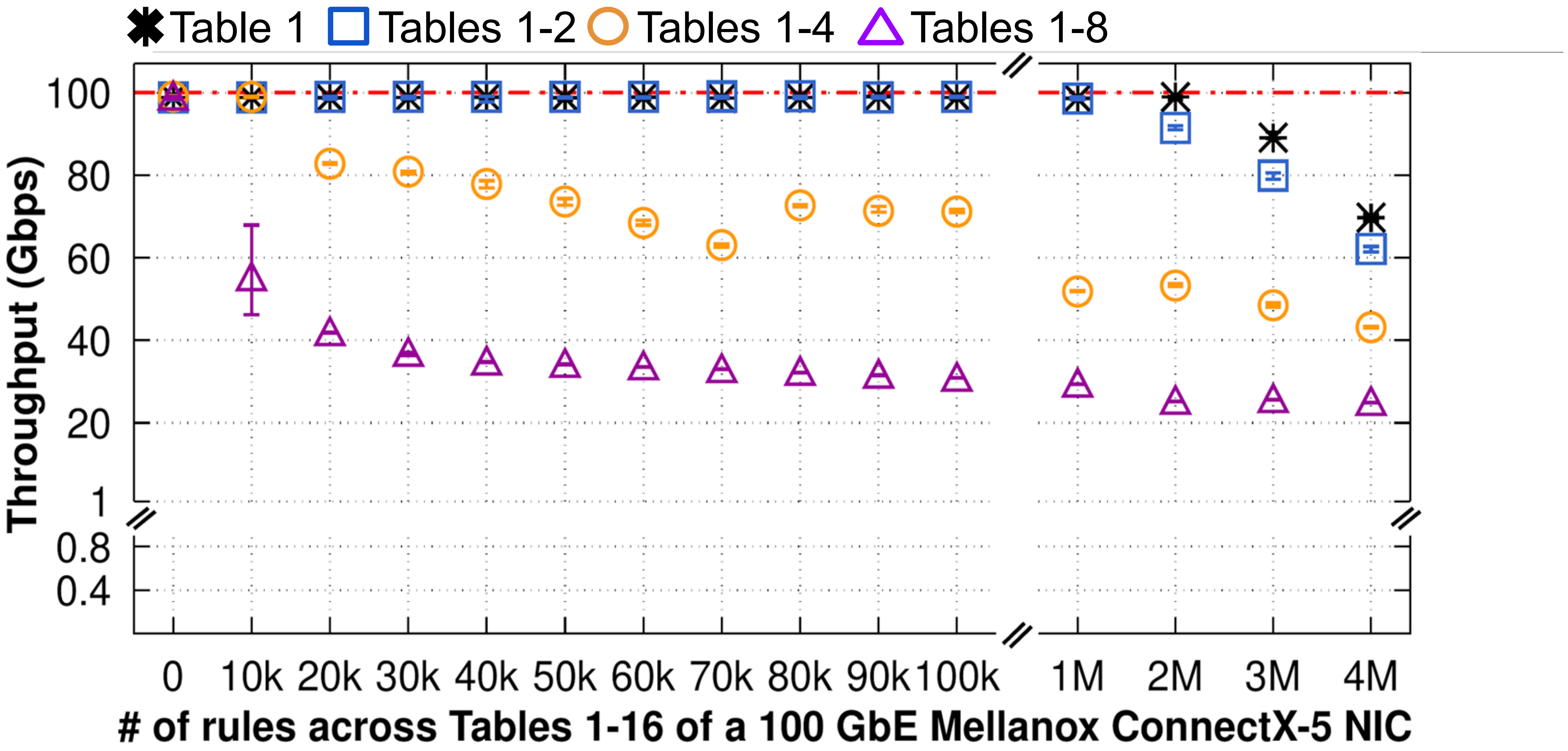
Throughput acc. number of tables



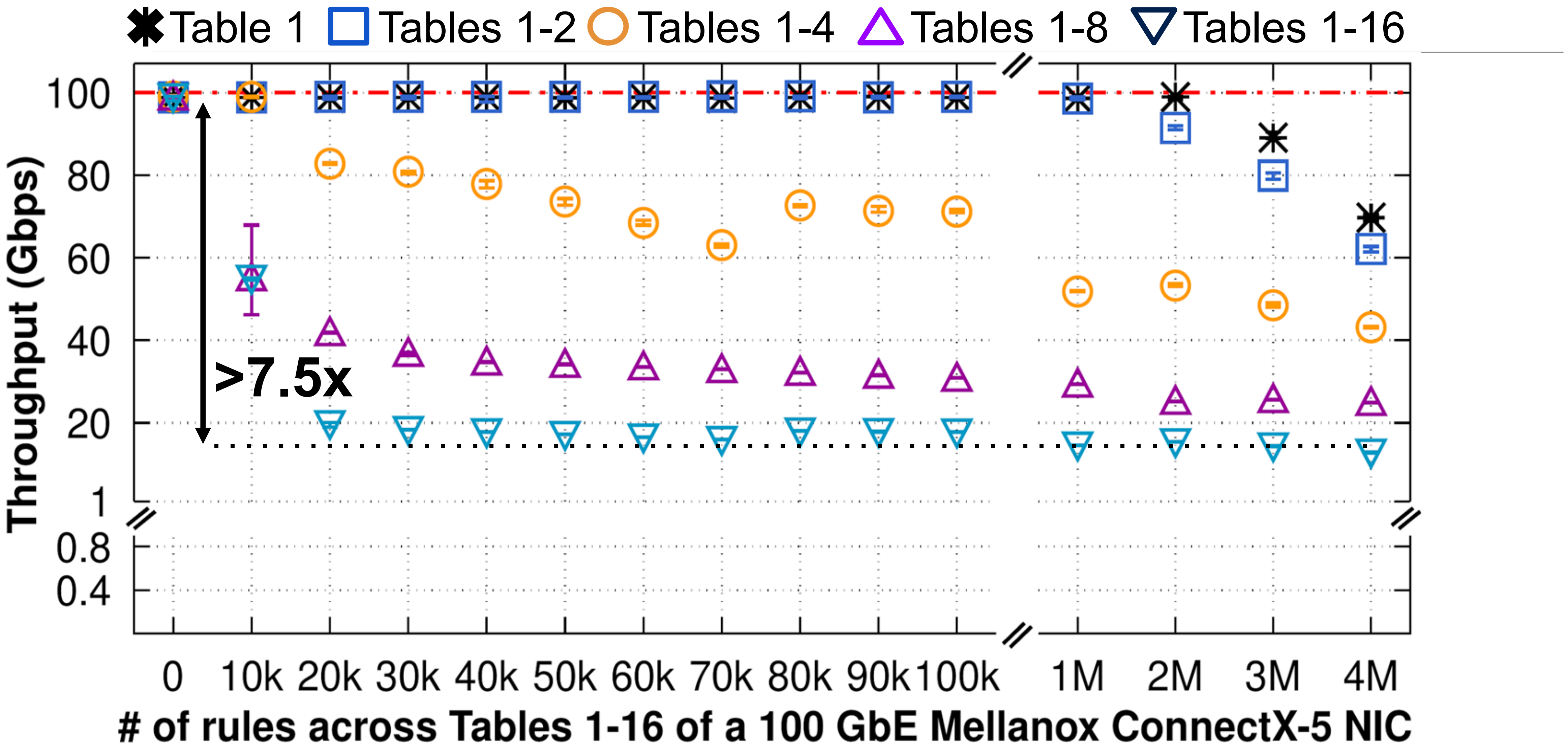
Throughput acc. number of tables



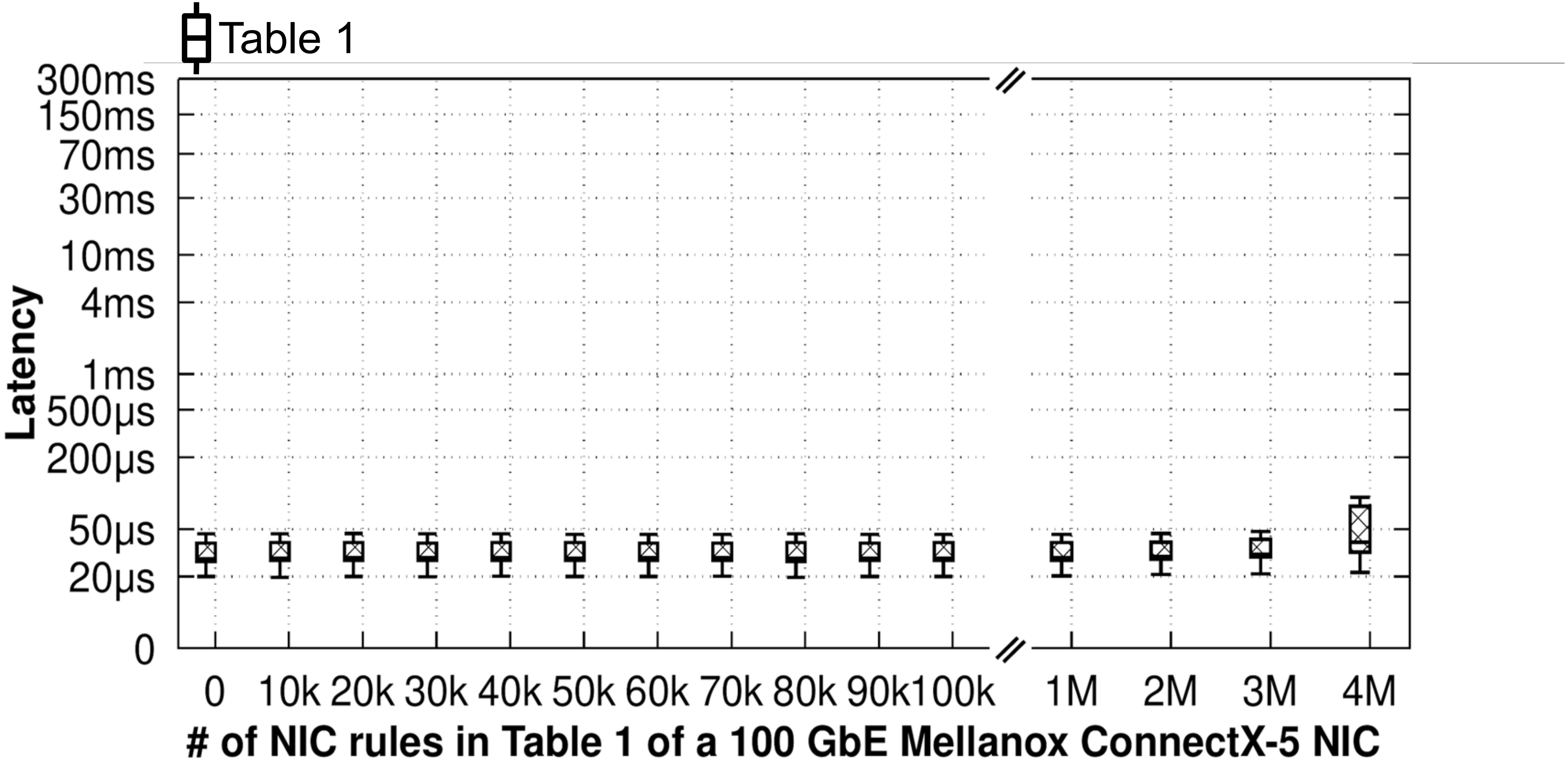
Throughput acc. number of tables



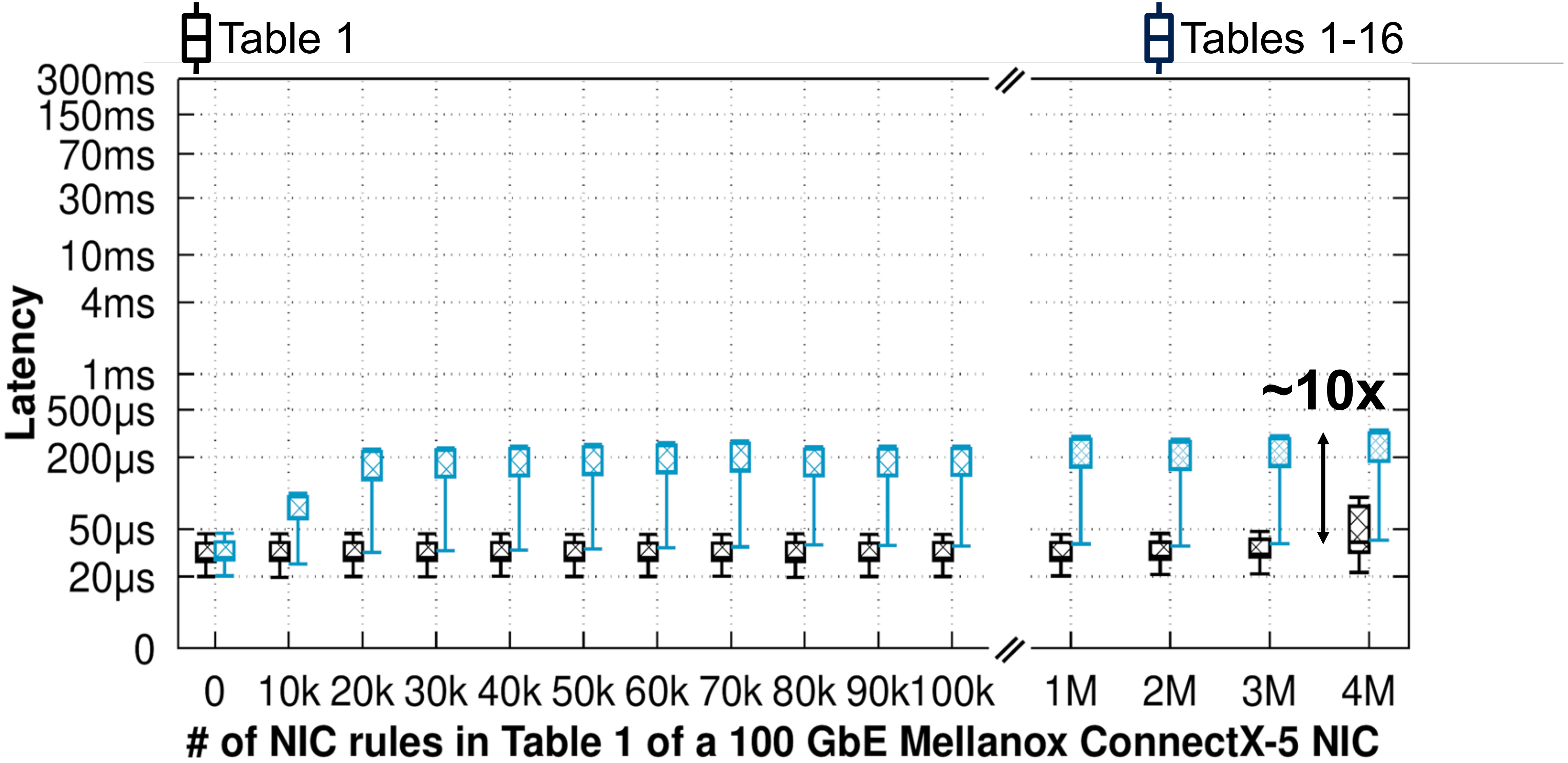
Throughput acc. number of tables



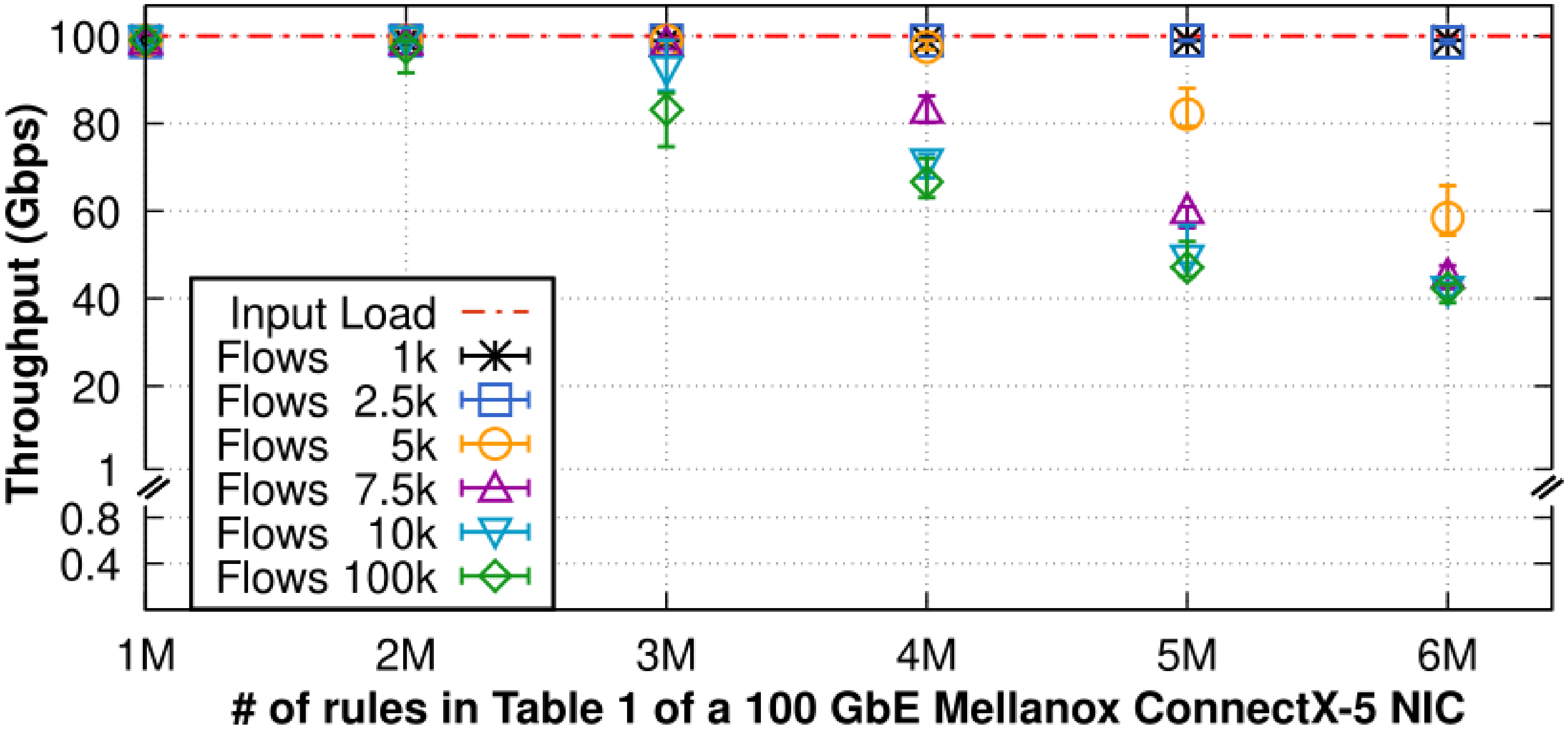
Scenario 1- Latency



Scenario 1- Latency



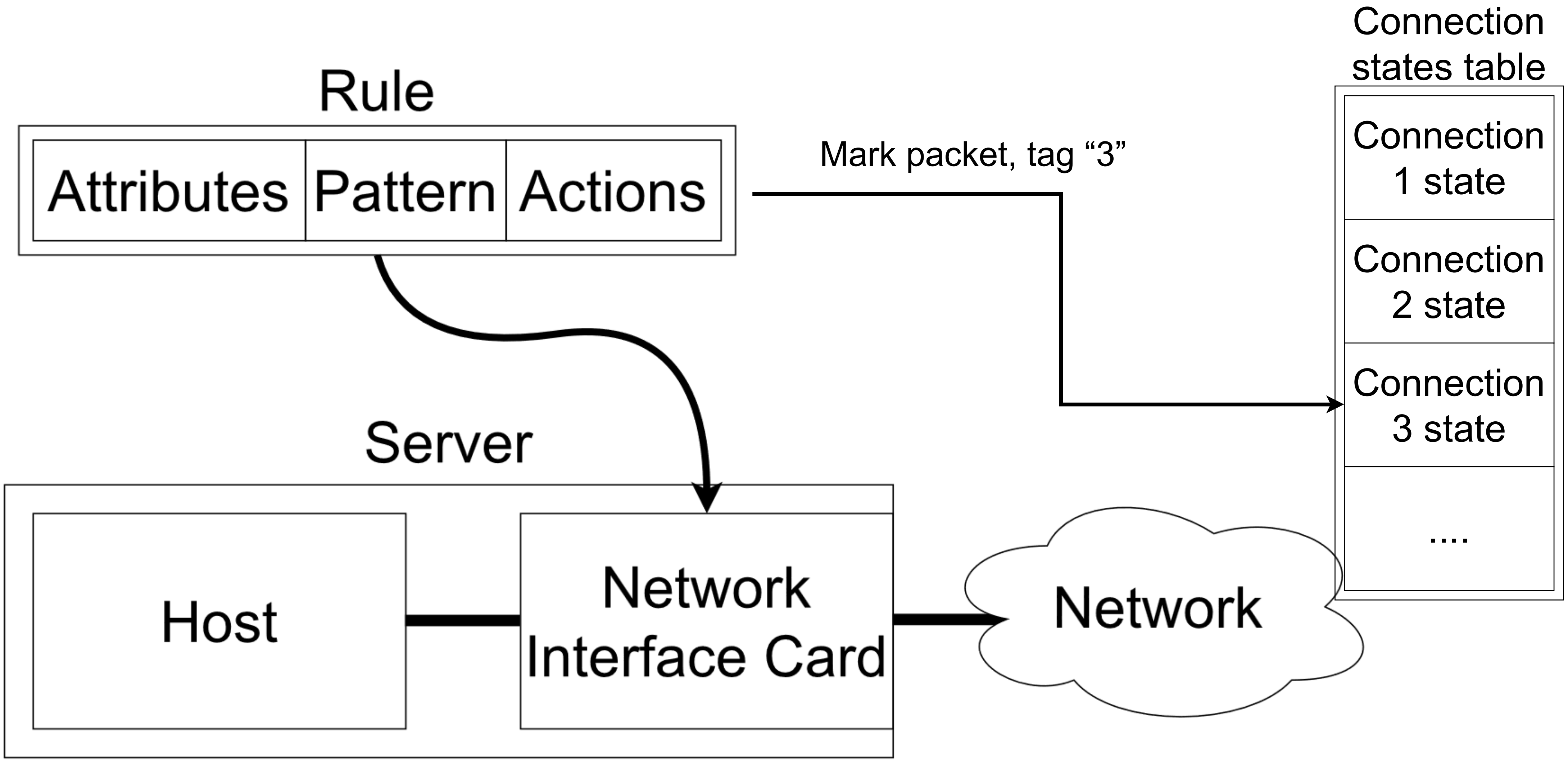
Impact of number of flows in the input load



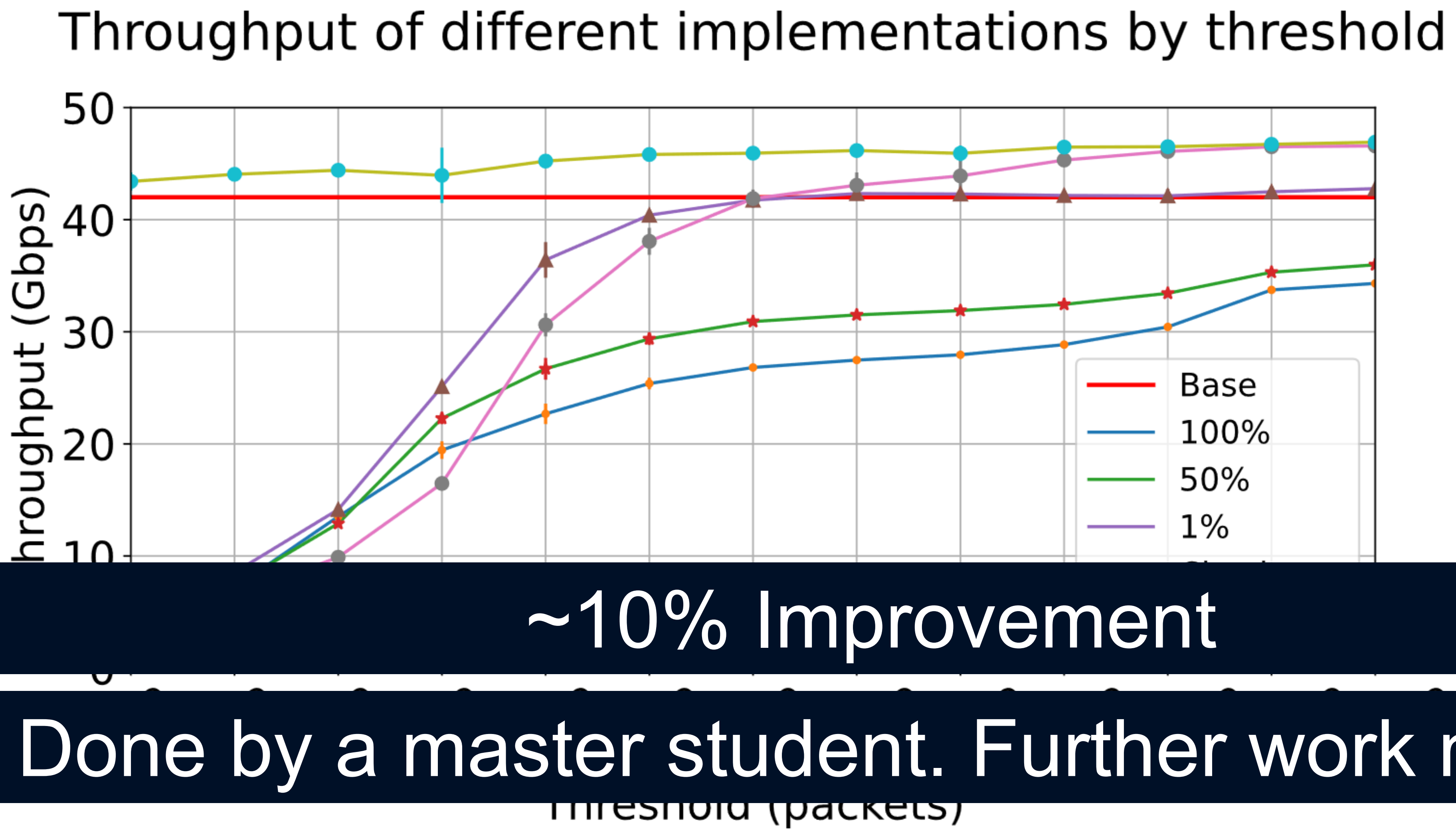
Offloading every flow is not going to work

Can we offload some of the flows?

How do we use the NIC to pre-process packets ?



WIP : Connection Tracking Offloading



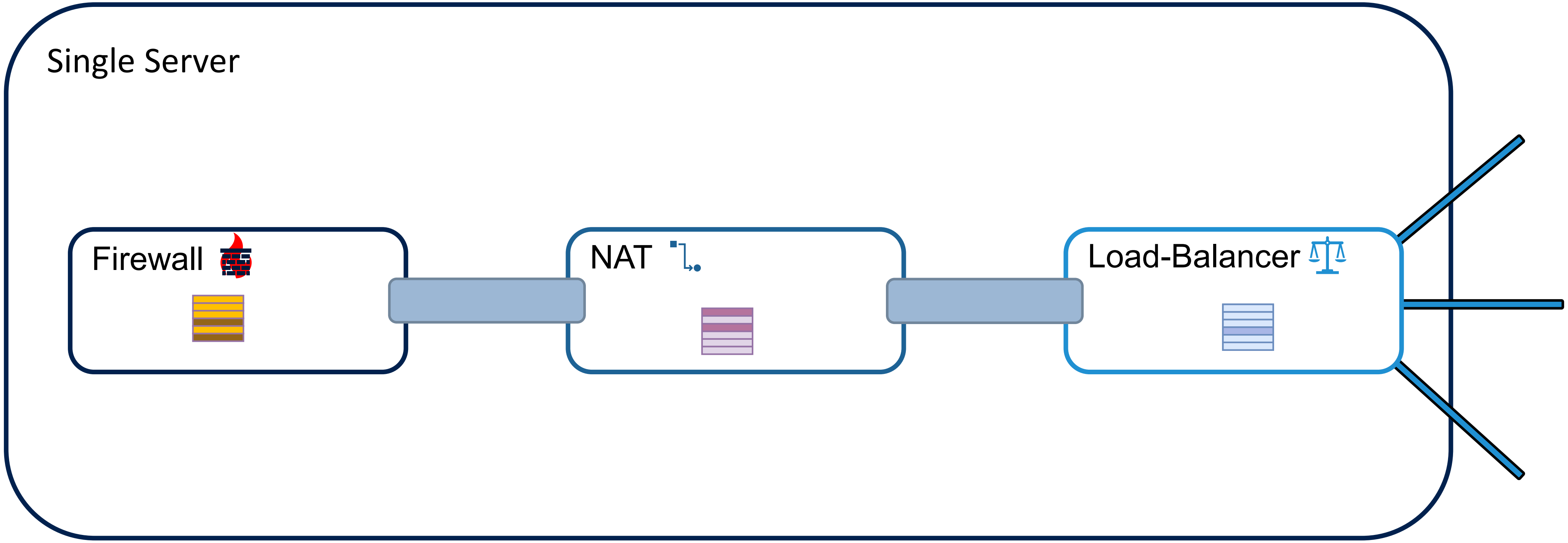
~10% Improvement

Done by a master student. Further work needed!

UCLouvain

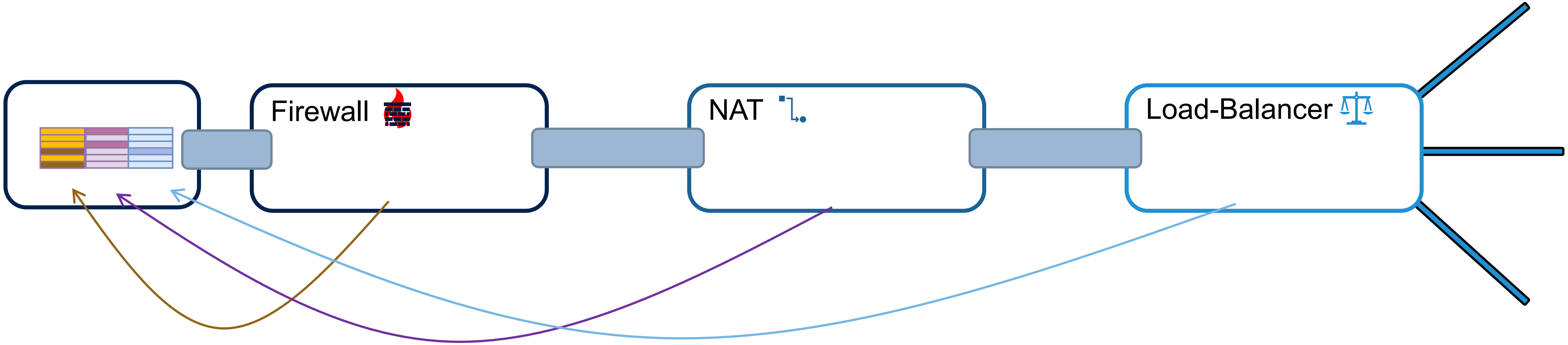
Can we combine the tracking for multiple Network Functions and offload some part of the classification ?

Service Chaining: Tracking inside each NF



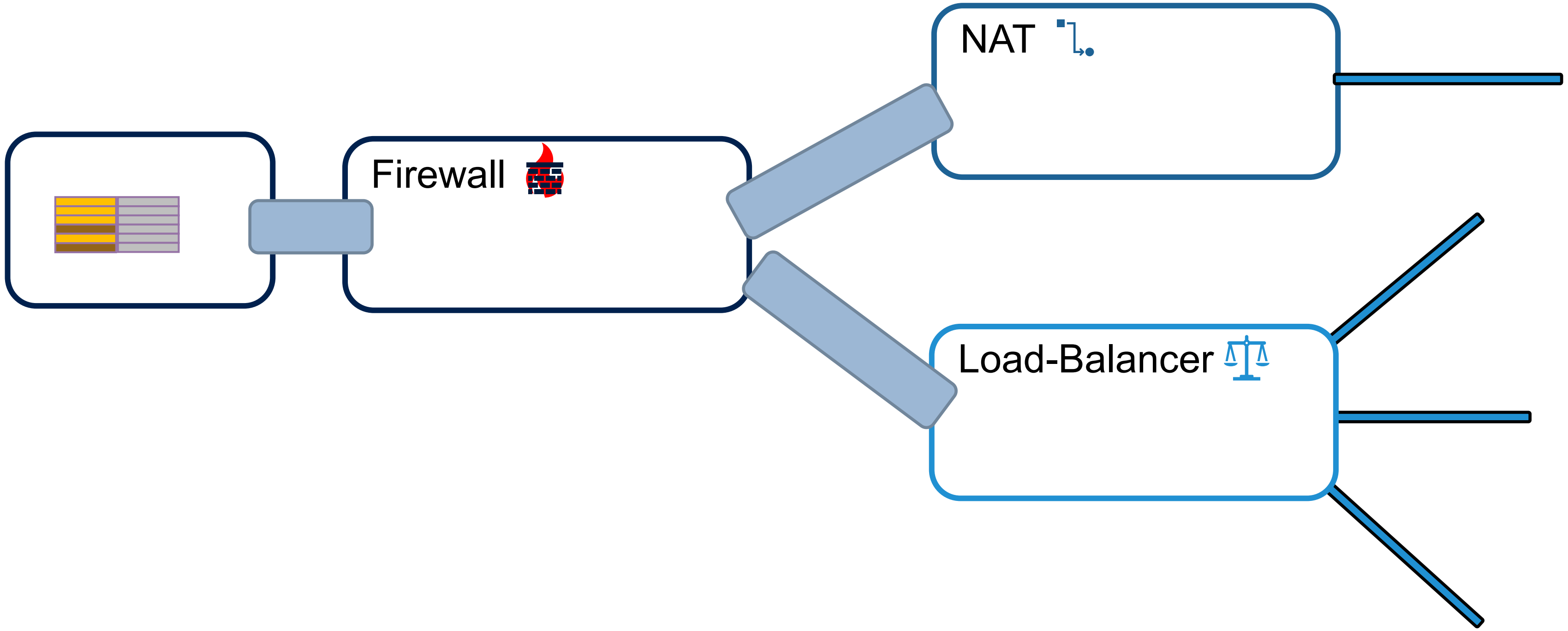
MiddleClick: Combining Classification

[IEEE/ACM ToN, 2021] T. Barbette, C. Soldani, L. Mathy



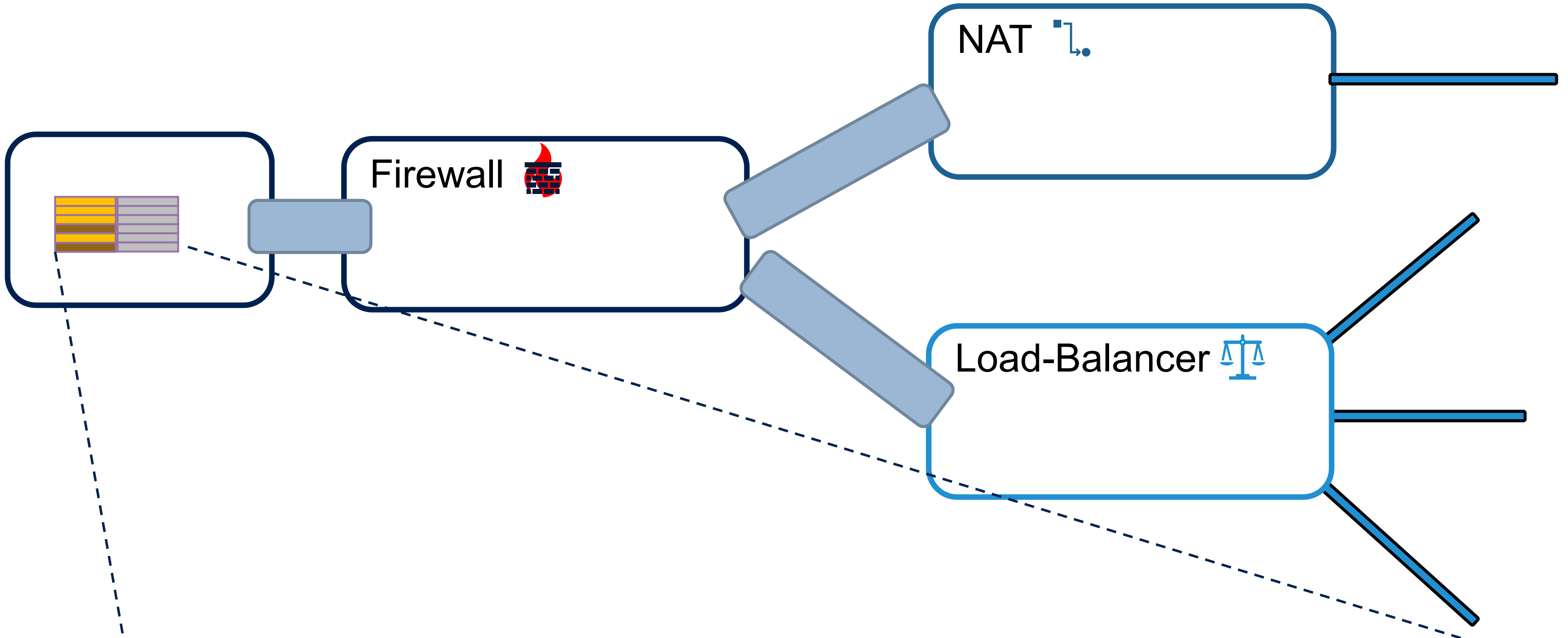
MiddleClick: Combining Classification

[IEEE/ACM ToN, 2021] T. Barbette, C. Soldani, L. Mathy

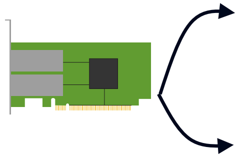


MiddleClick: Combining Classification

[IEEE/ACM ToN, 2021] T. Barbette, C. Soldani, L. Mathy

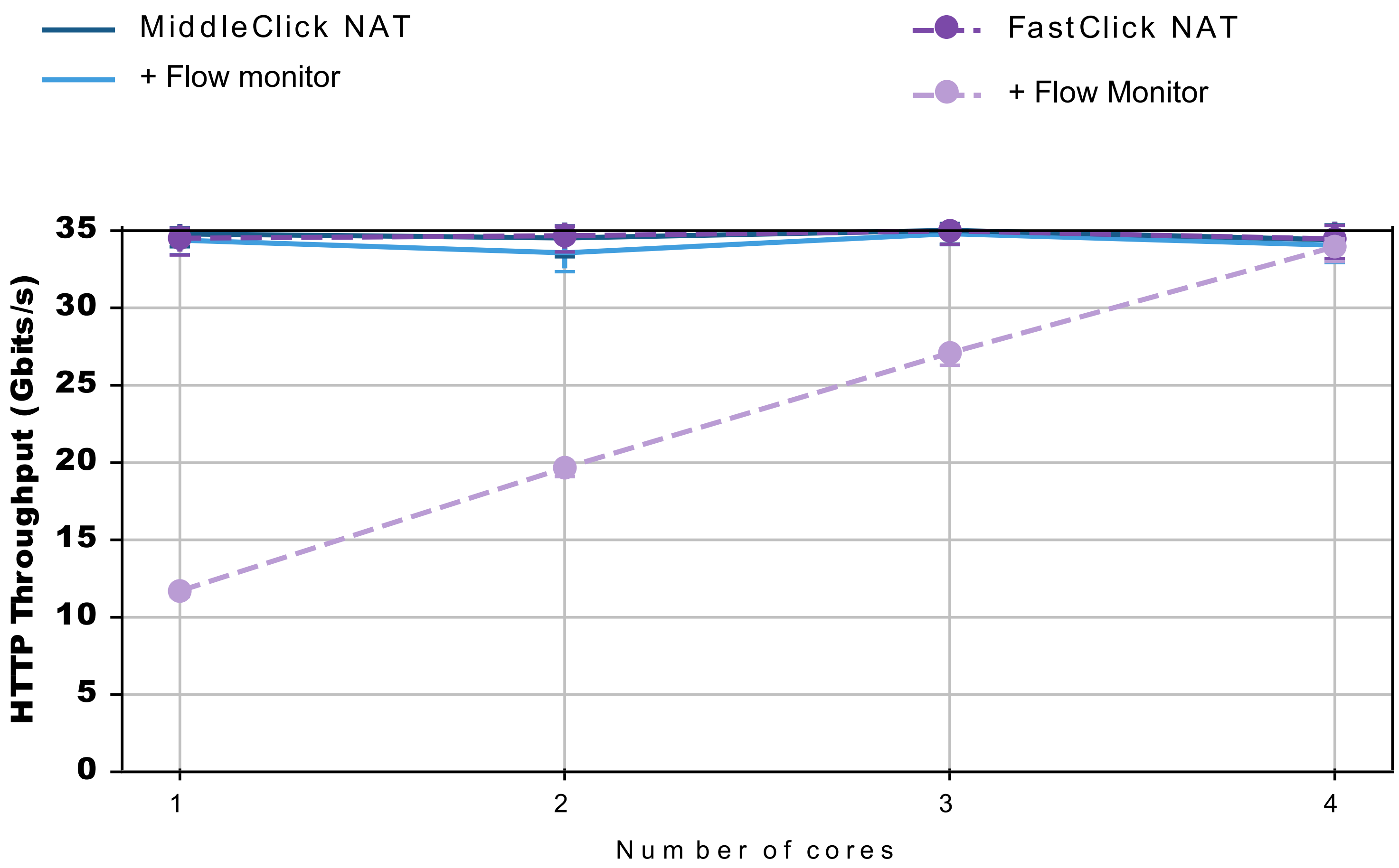


Flow Control Block



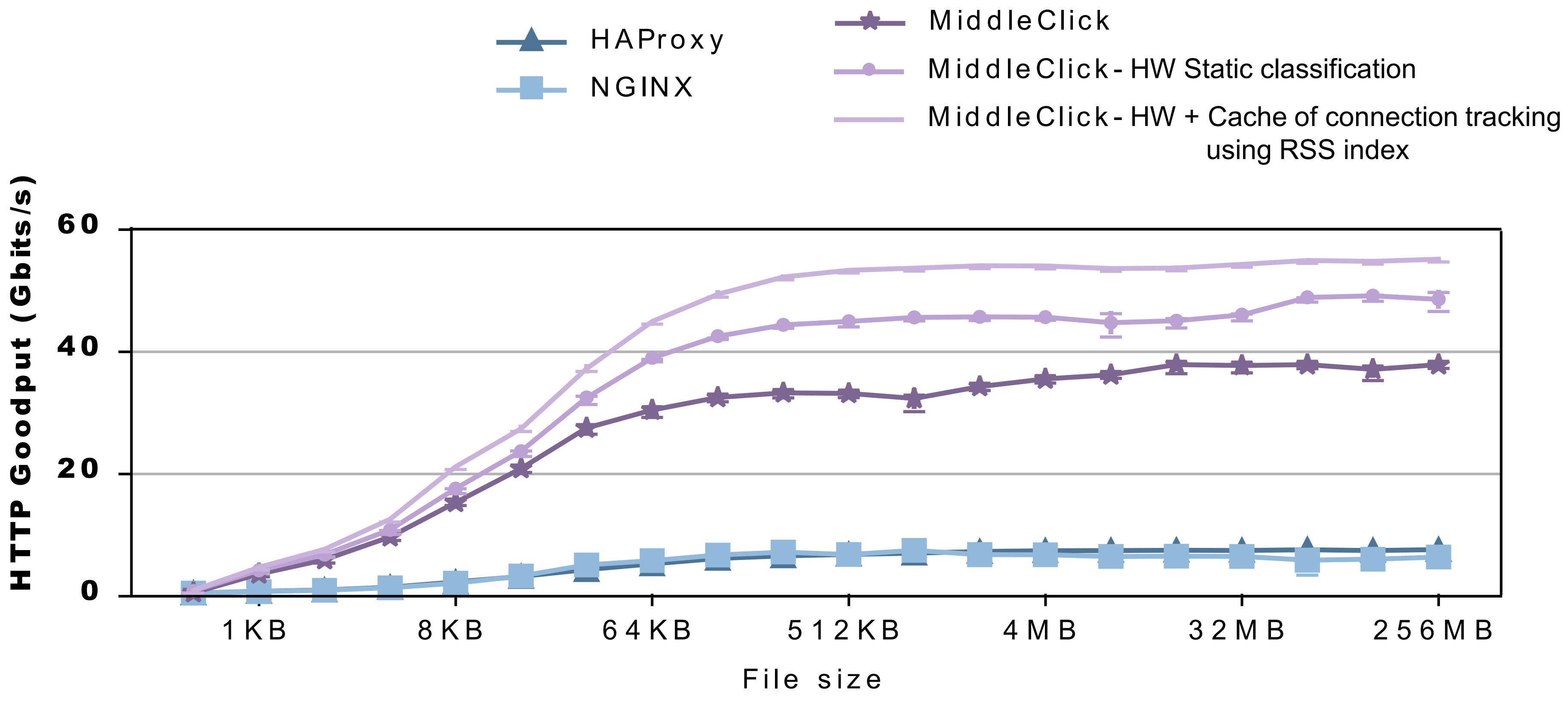
Metadata	Firewall	NAT	
Metadata	Firewall	Load-balancer	

Evaluation : Avoiding re-classification



NAT running between 128 HTTP clients making requests to 4 NGINX servers – 35Gbps is the limit of the testbed

Evaluation: Offloading (HTTP load-balancer)



UCLouvain

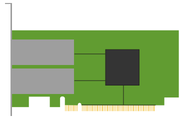
Takeout

- High-speed stateful software packet processing
  - Sharding is the way to go
  - Especially with the trend to many cores
  - Load-balancing problem : use RSS++ [CoNEXT'19]
- Switch-based stateful packet processing
  - Ribosome processes 300Gbps worth of traffic with 20Gbps of bandwidth
  - Send what you need where you actually need it
- NIC-assisted stateful packet processing
  - Promising approach, still under development
- Combined stateful packet processing
  - Do not re-classify the same thing
  - Hardware does help !

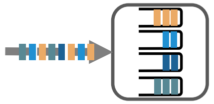
Conclusion

www.tombarbette.be

tom.barbette@uclouvain.be



- **High-speed** packet processing techniques
  - NFV with FastClick (+PacketMill + MiddleClick + RSS++)µ
  - Integration for software state (ConnTrack + MiddleClick)
- **Load-balancing**
  - Inside (RSS++) and between servers (Cheetah) or both (Metron, CrossRSS)
- The **network is starting to be programmable**, and has per-connection programmability
  - Job scheduling, optimization
- **Build the infrastructure for an efficient, competitive, and local Internet**
  - Make the network's core programmable by the service provider
  - Improve today's network efficiency and enable the agility needed to sustain tomorrow's services



Try FastClick, a high speed dataplane based on Click and its PacketMill improvement ! While load-balacing with RSS++ and combining sessions with MiddleClick, all included!

Try Retina for high-speed passive traffic analysis!

UCLouvain