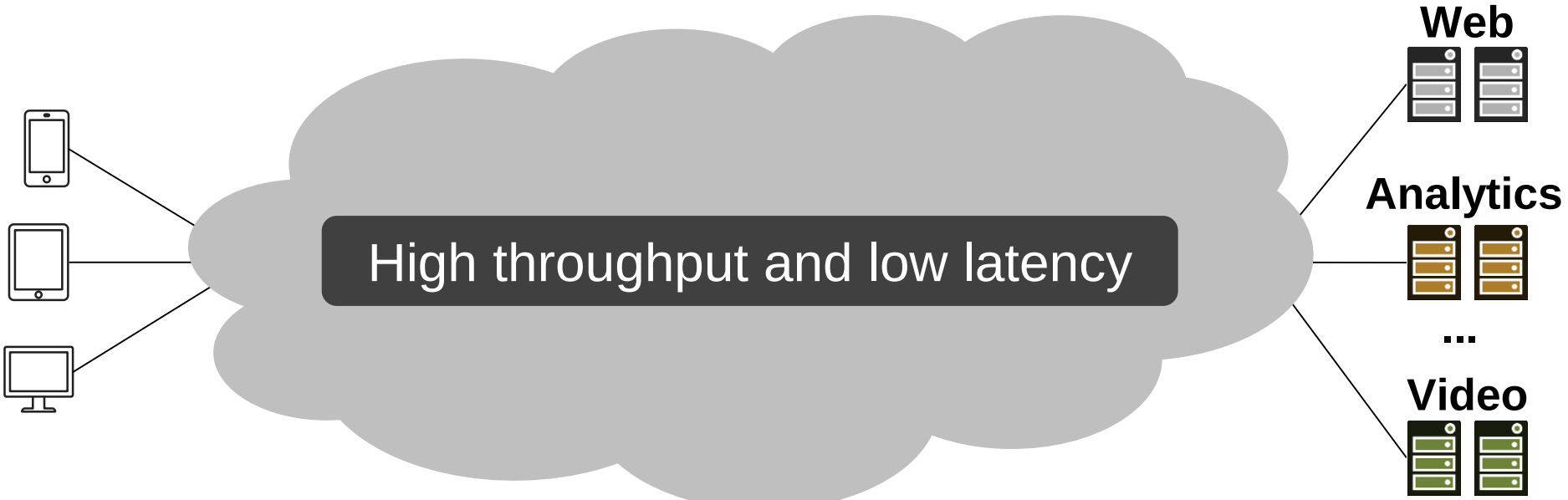
# Metron
# NFV Service Chains at the True Speed of the Underlying Hardware

Georgios P. Katsikas[1,3], Tom Barbette[2], Dejan Kostic[3],
Rebecca Steinert[1], and Gerald Q. Maguire Jr.[3]

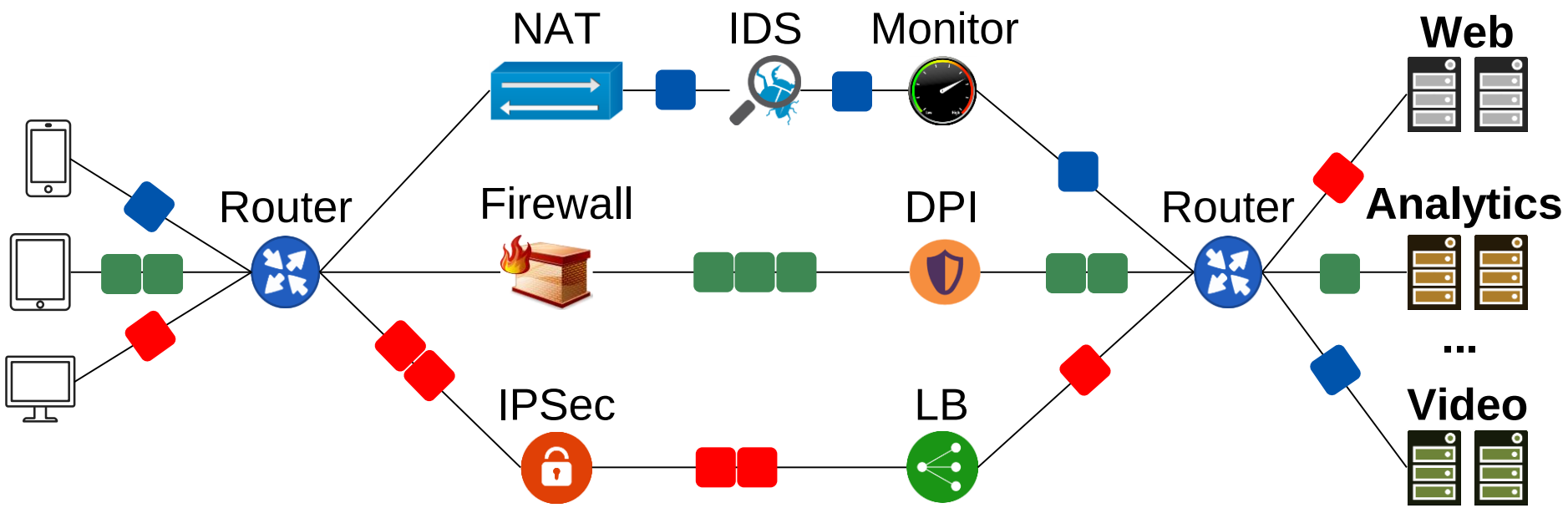
[1]RISE SICS [2]University of Liege [3]KTH Royal Institute of Technology



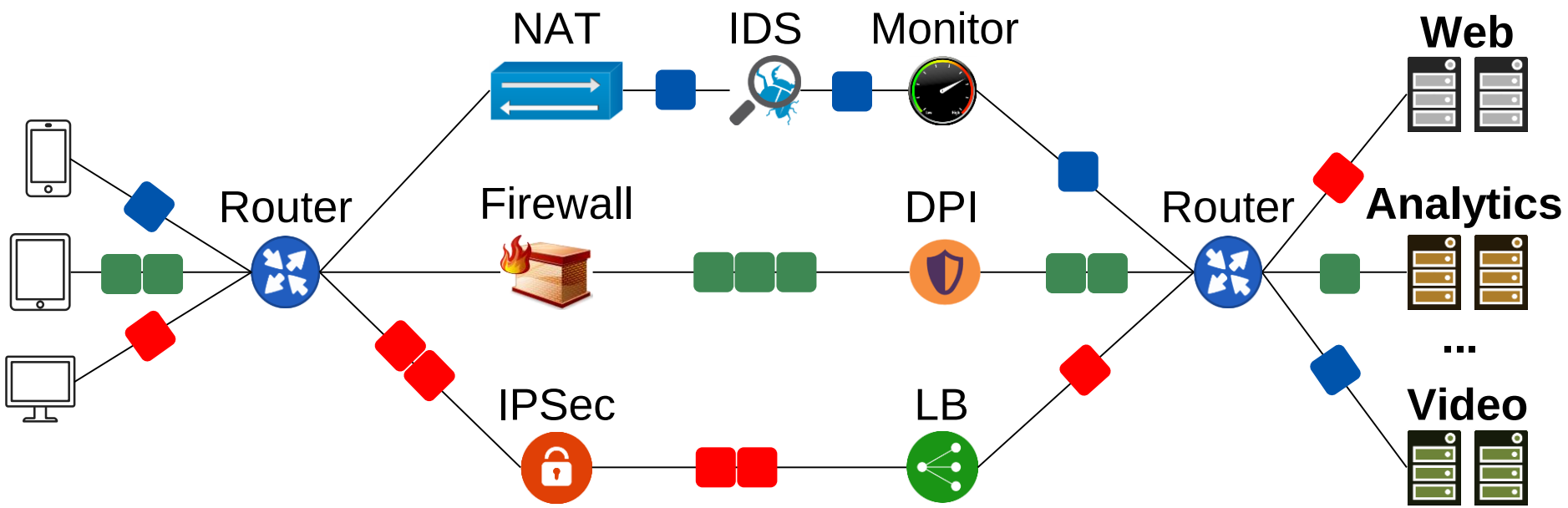
# Introduction



# Service Chains of Network Functions



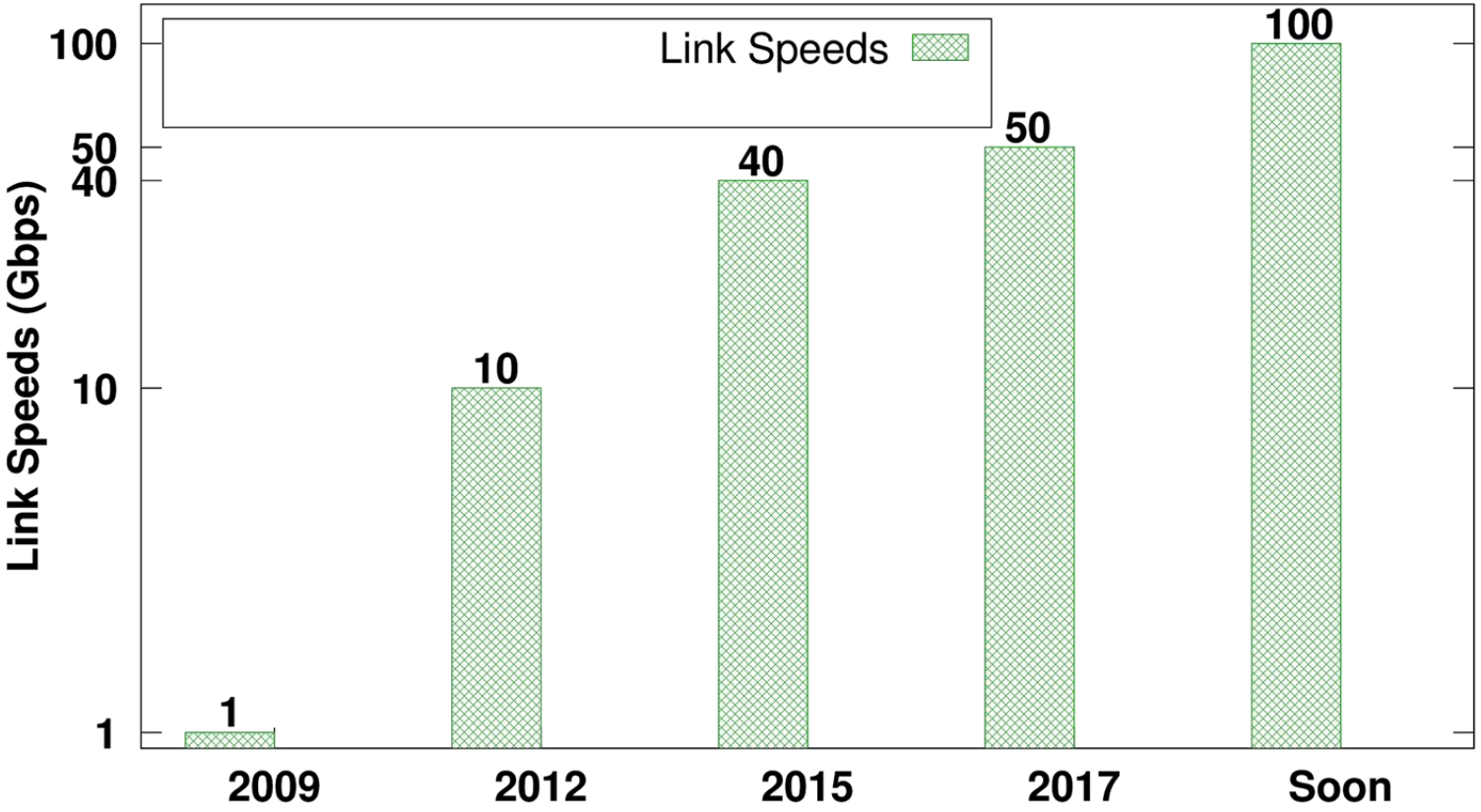
# Service Chains of Network Functions




Minimize the performance impact of service chains

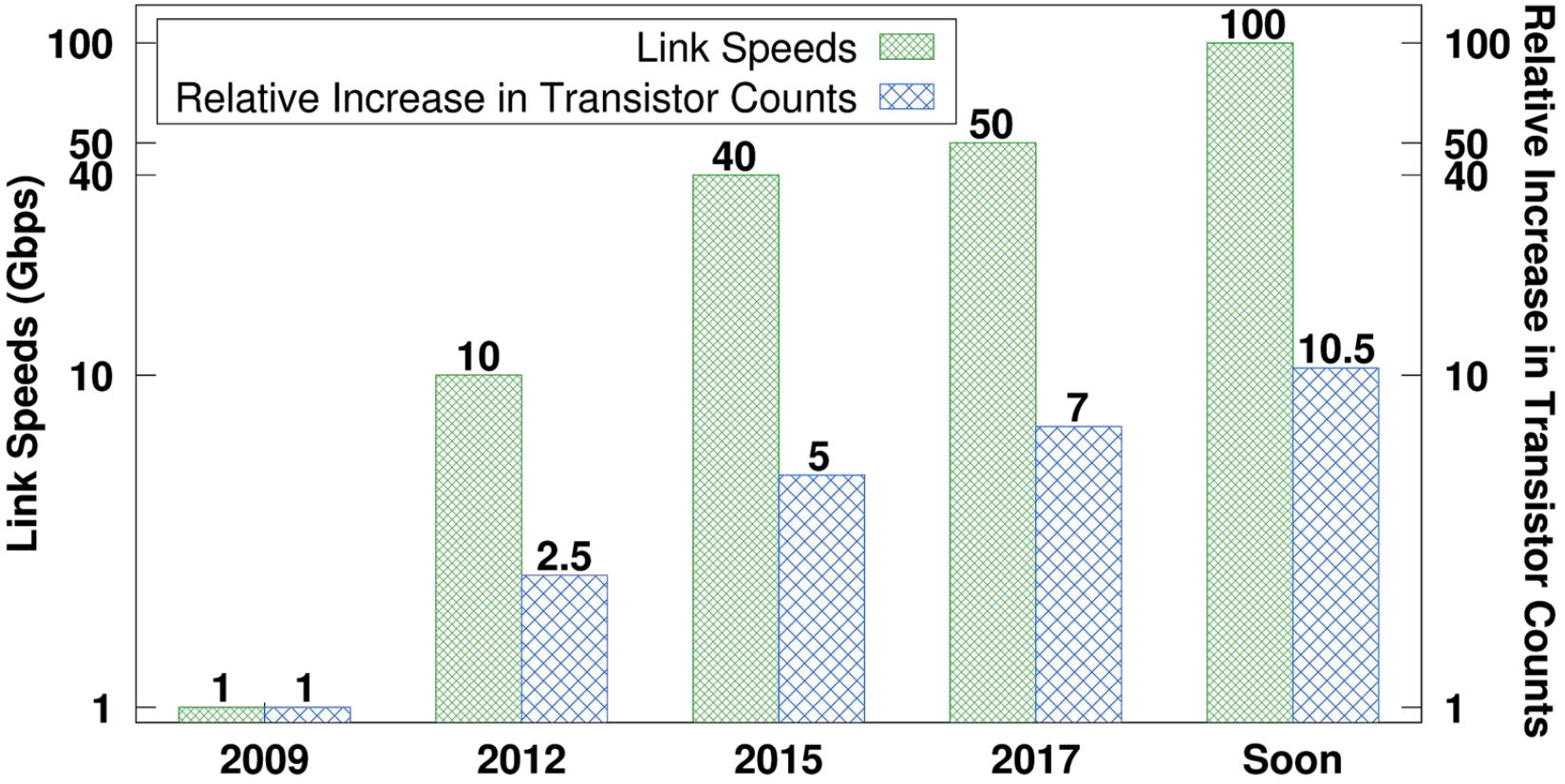
# Motivation

Source: D. Firestone et al. Hardware-Accelerated Networks at Scale in the Cloud, Microsoft Azure, SIGCOMM 2017



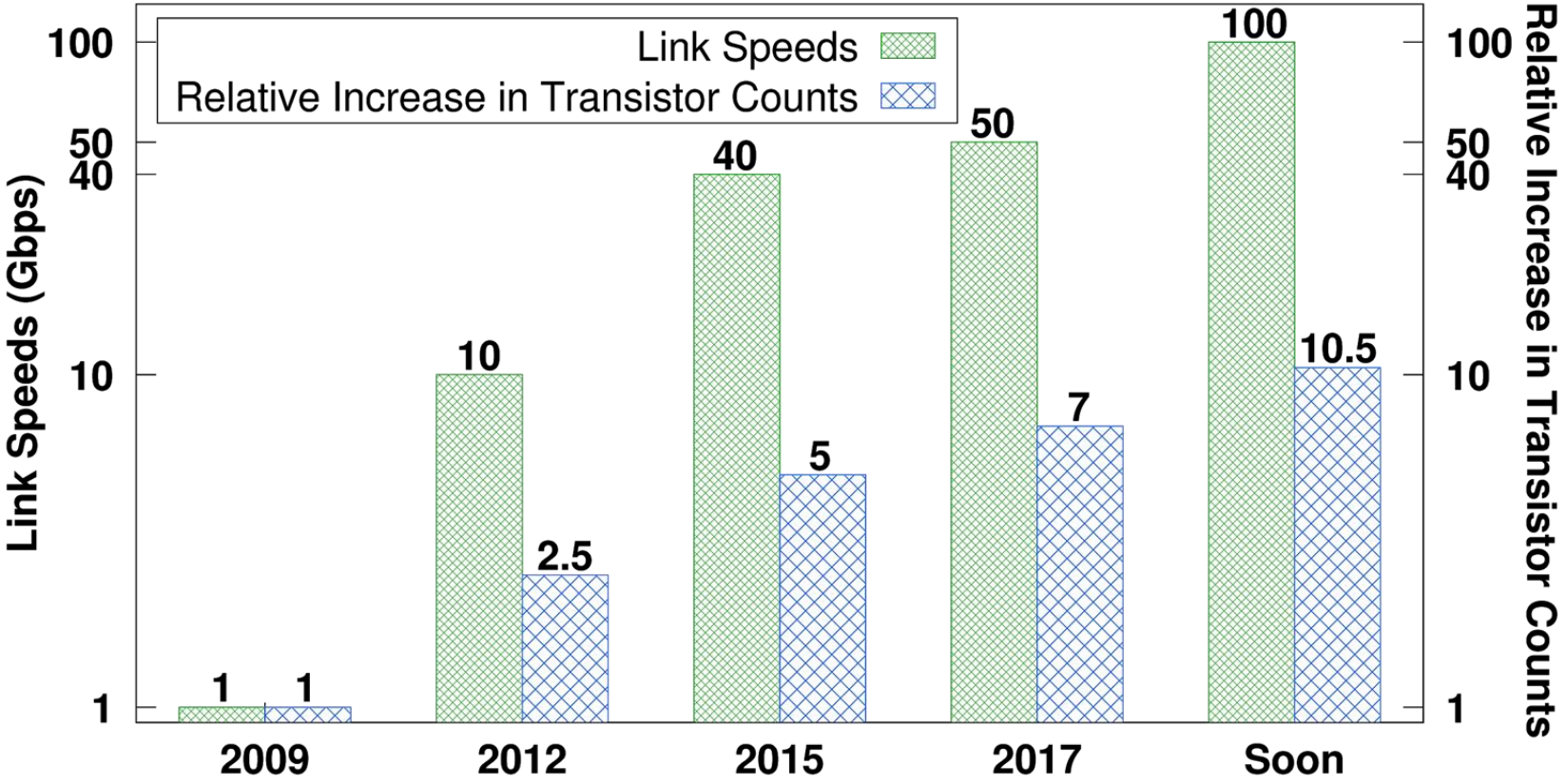
# Motivation

Source: D. Firestone et al. Hardware-Accelerated Networks at Scale in the Cloud, Microsoft Azure, SIGCOMM 2017



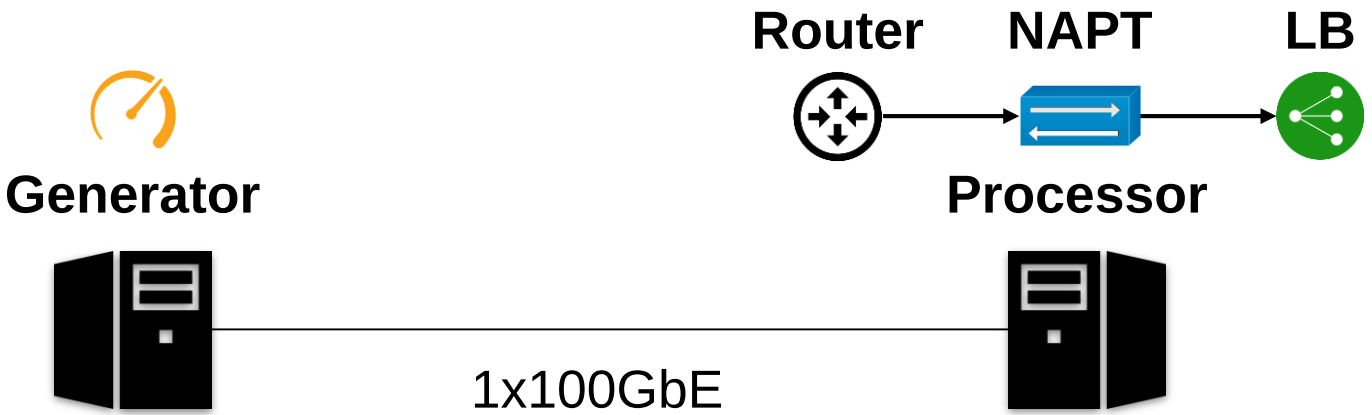
# Motivation

**Service chains today greatly rely on CPU performance!**



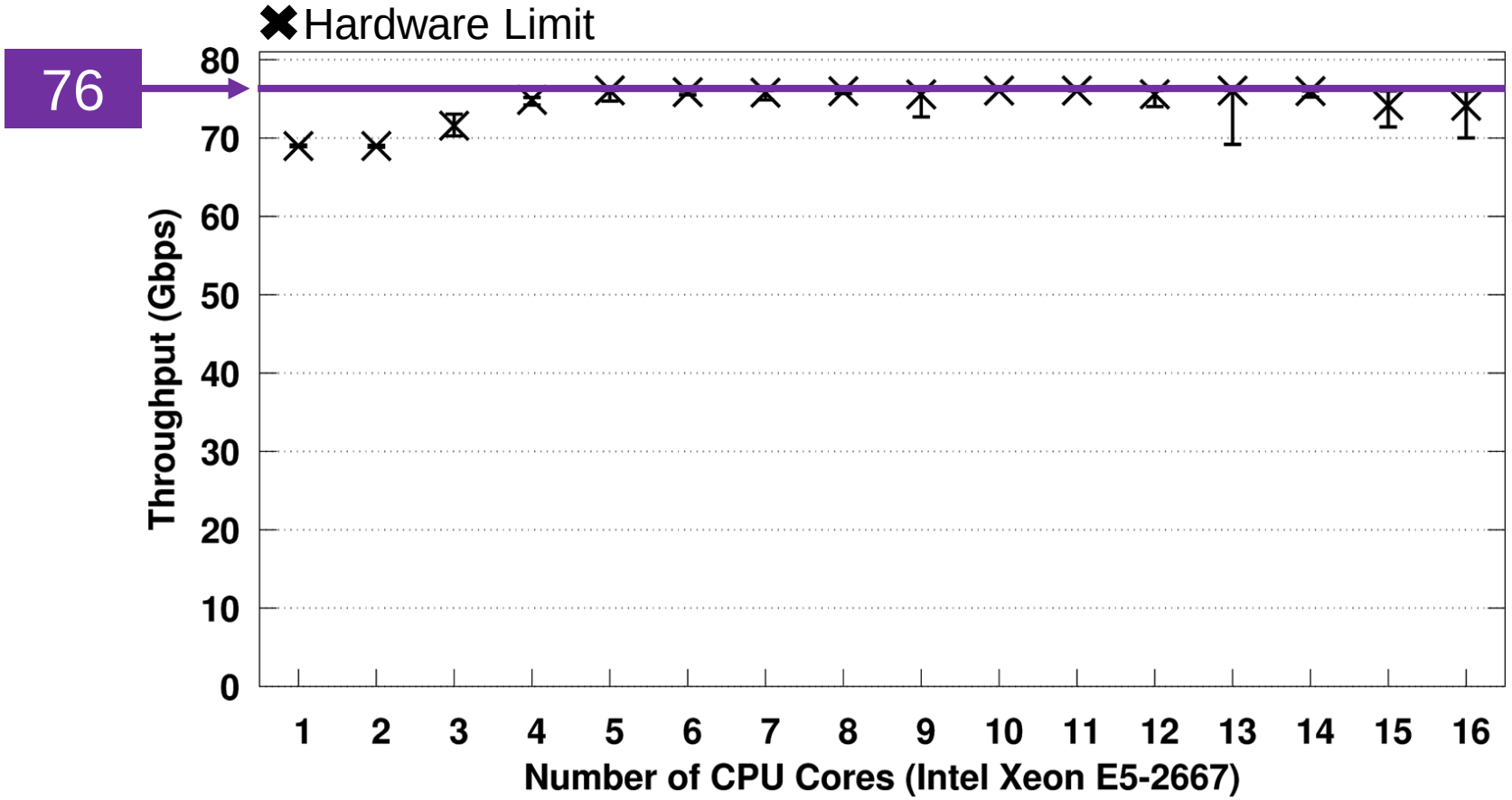
Can existing NFV systems operate at these emerging link speeds (i.e., at 100 Gbps)?

# Deployment at 100 Gbps

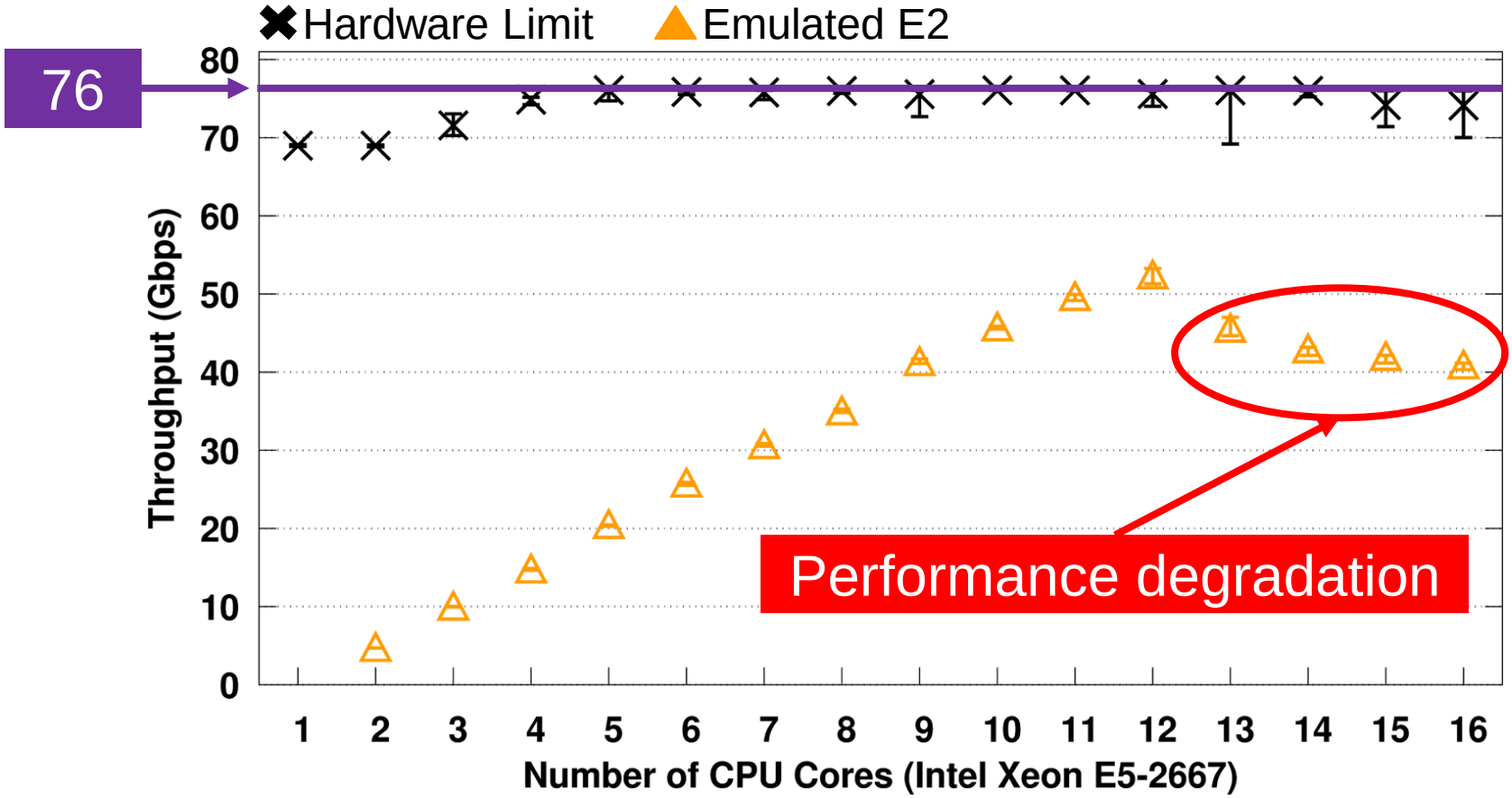



Total Throughput
100 Gbps

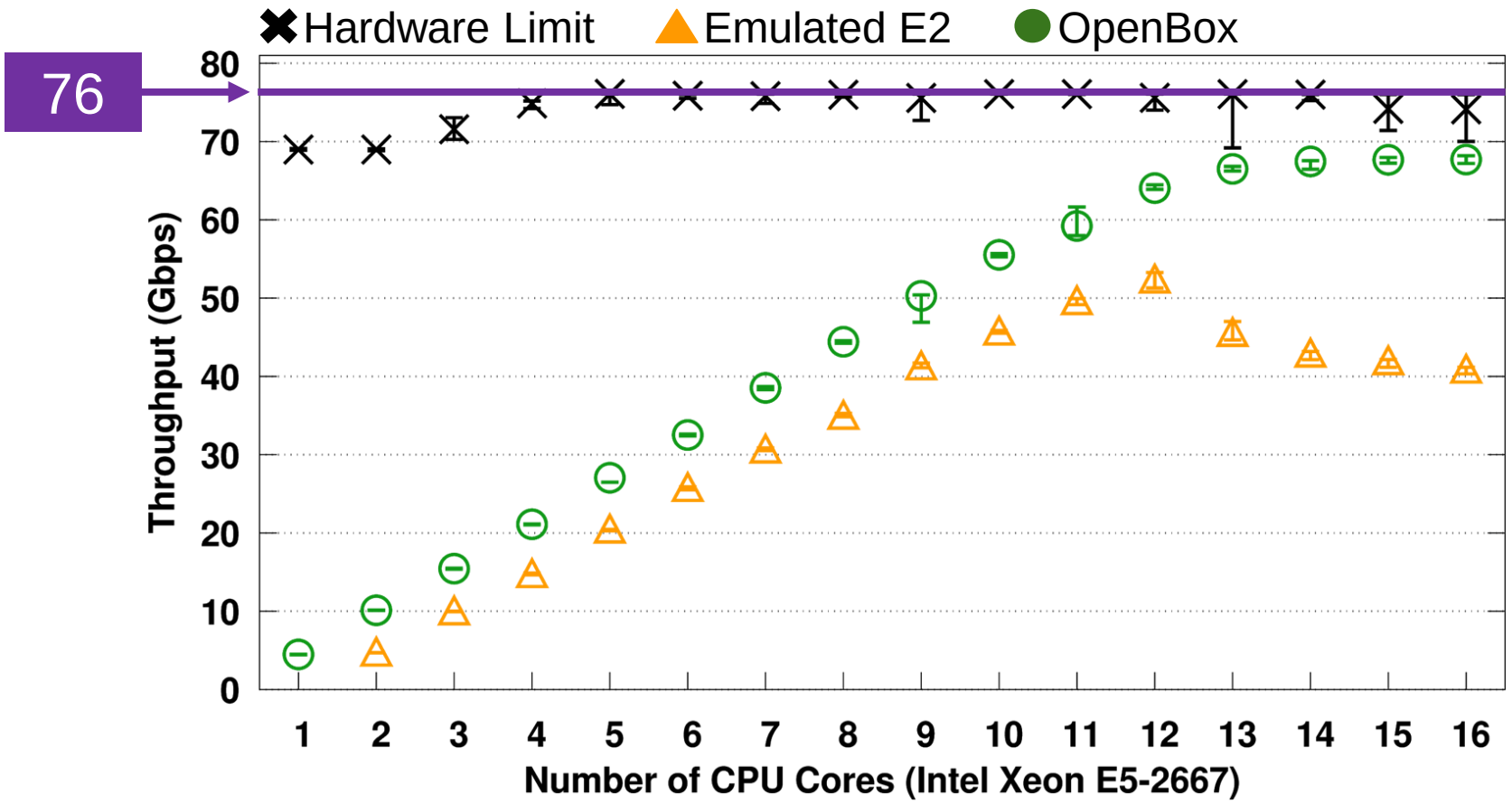
# Hardware Limit at 100 Gbps



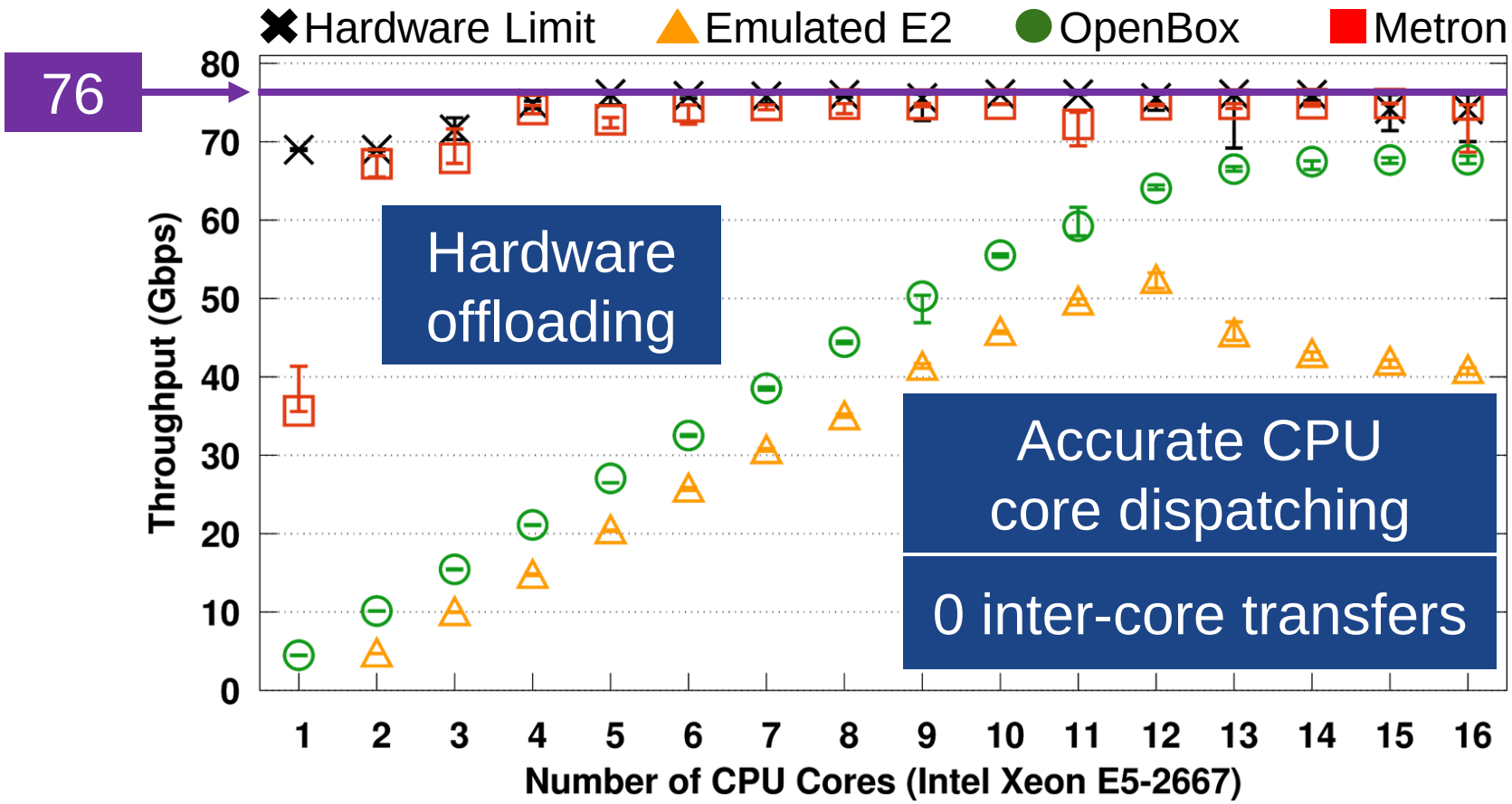
# Existing NFV Solutions at 100 Gbps



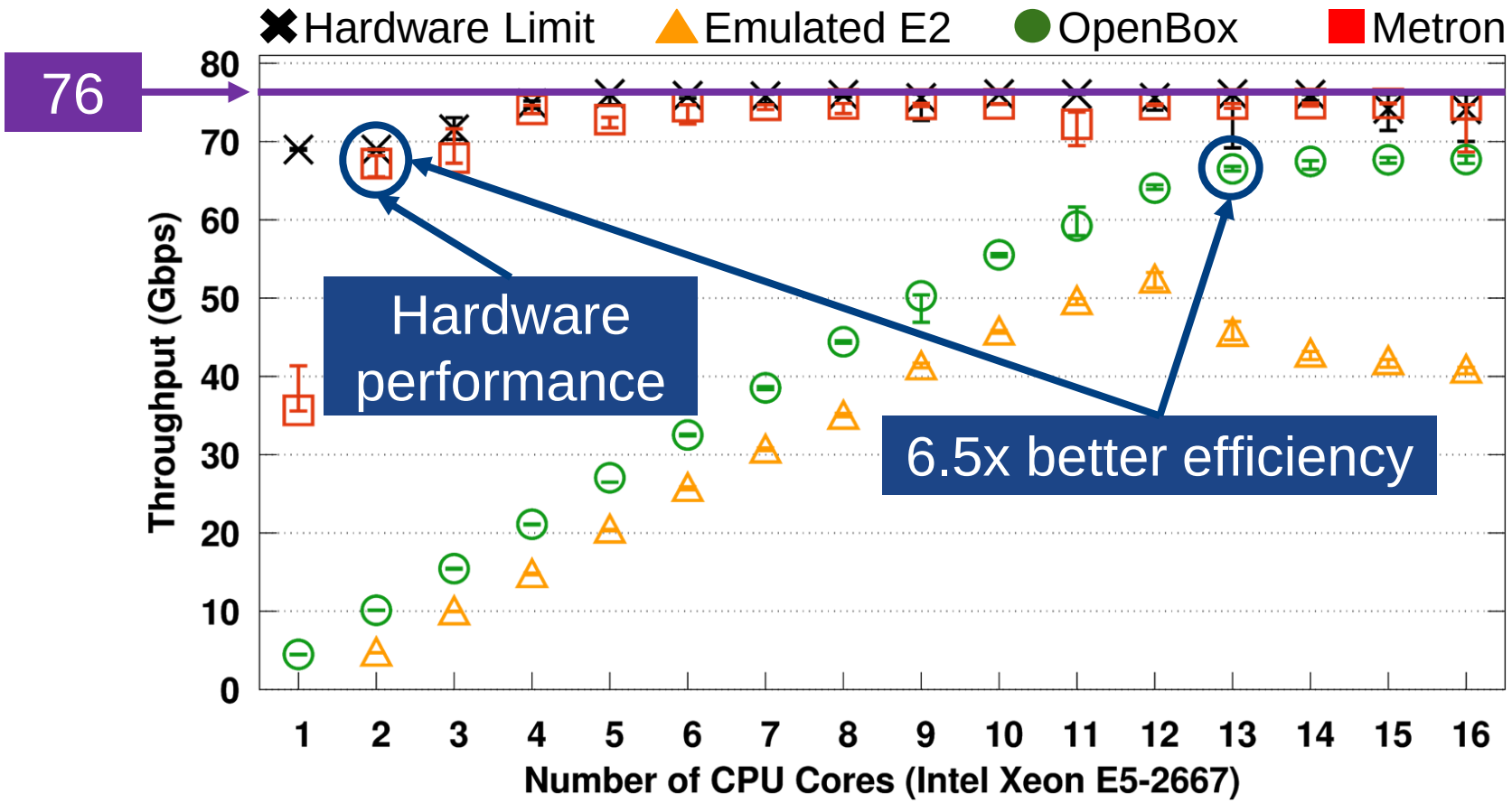
# Existing NFV Solutions at 100 Gbps



# Metron – Hardware-level Performance

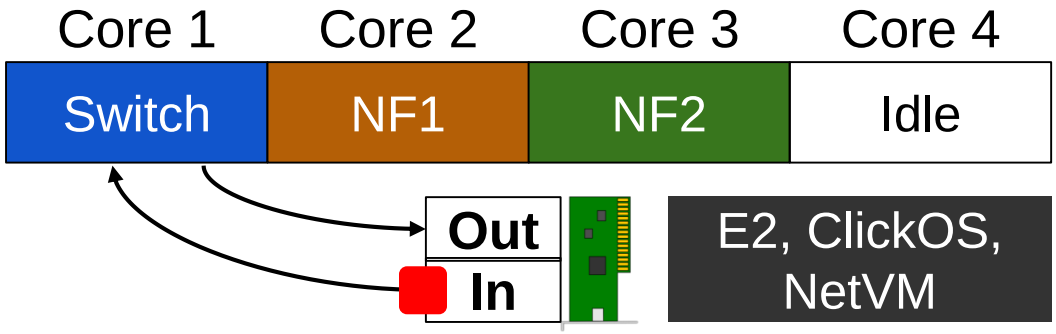


# Metron – Hardware-level Performance

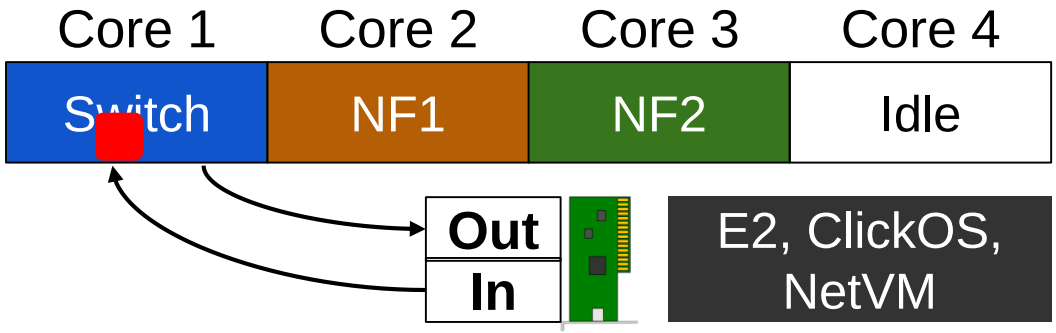


# Why do existing NFV systems lose performance?

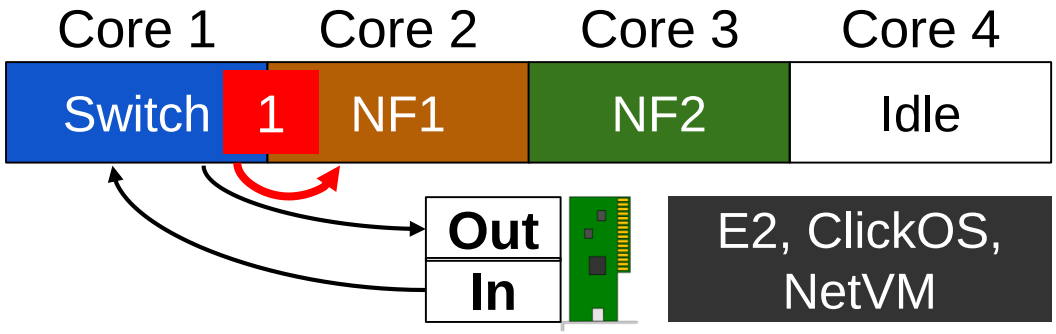
Software switch dispatching



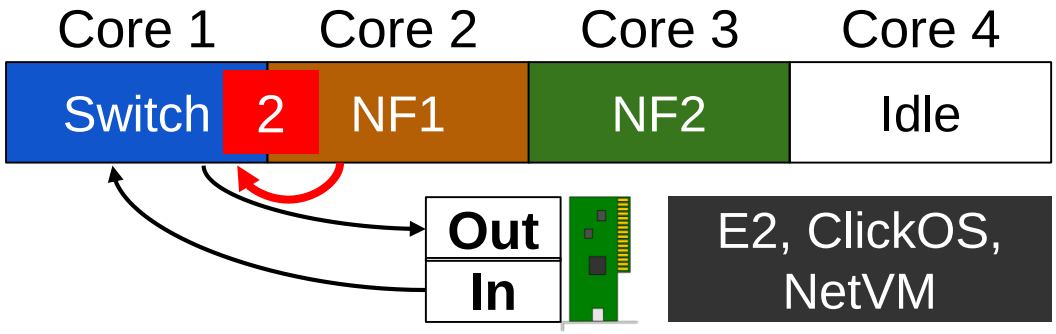
Software switch dispatching



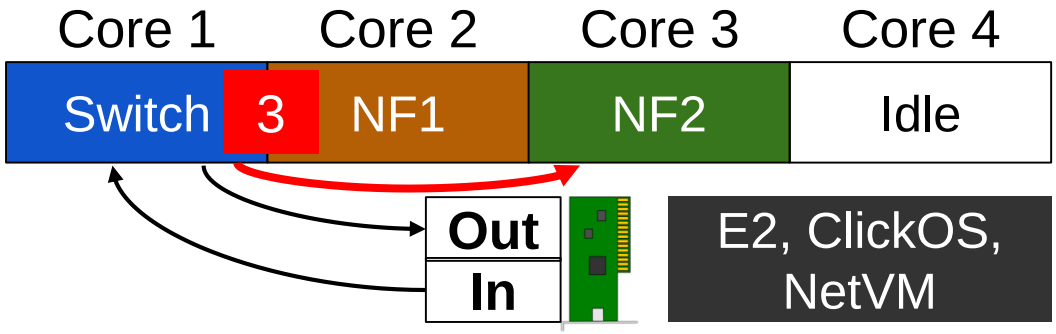
Software switch dispatching



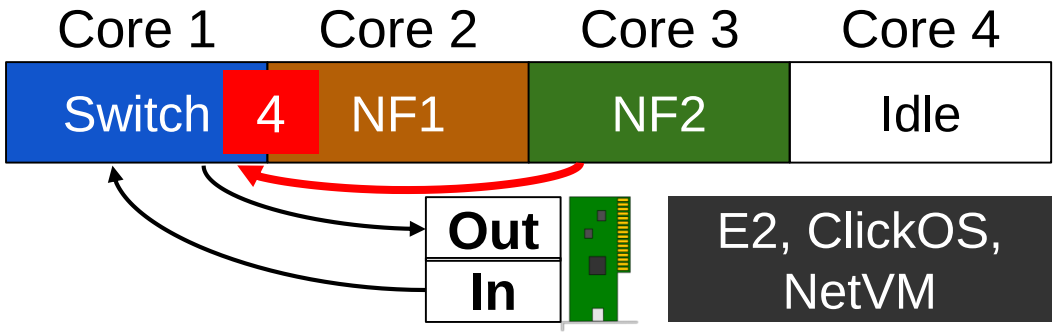
Software switch dispatching



Software switch dispatching

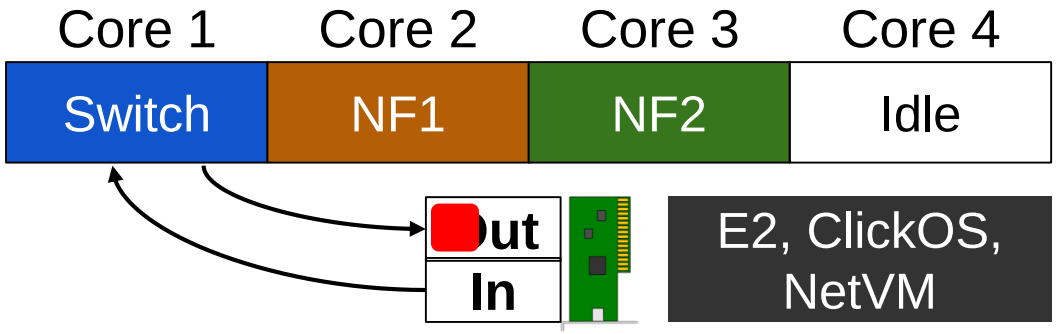


Software switch dispatching



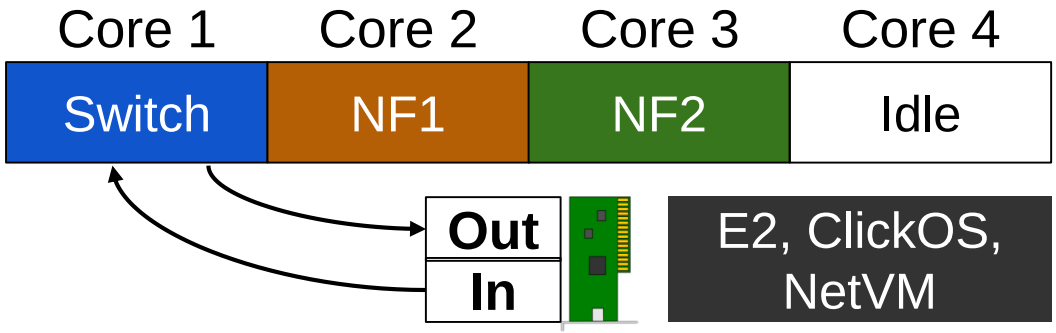
Software switch dispatching

4 Inter-Core Transfers

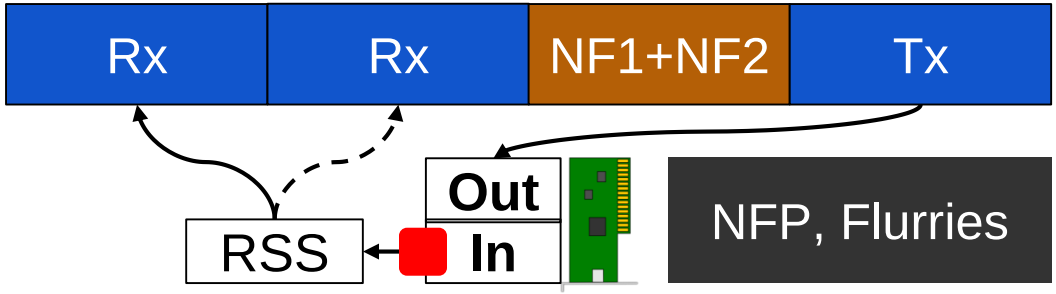


Software switch dispatching

4 Inter-Core Transfers

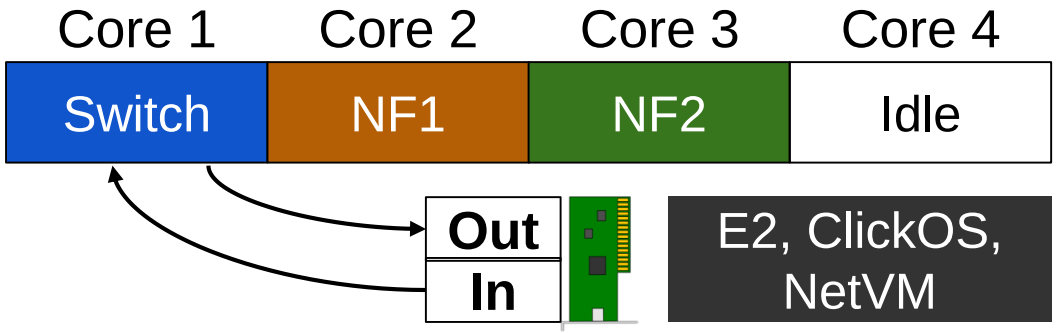



Pipeline dispatching with or without RSS

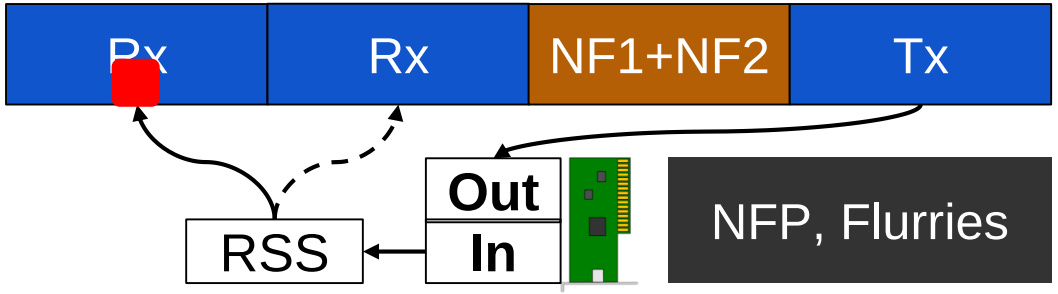


Software switch dispatching

4 Inter-Core Transfers

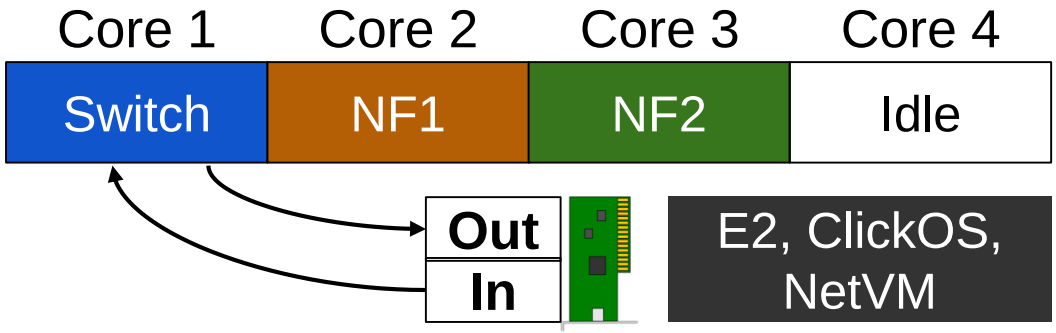



Pipeline dispatching with or without RSS

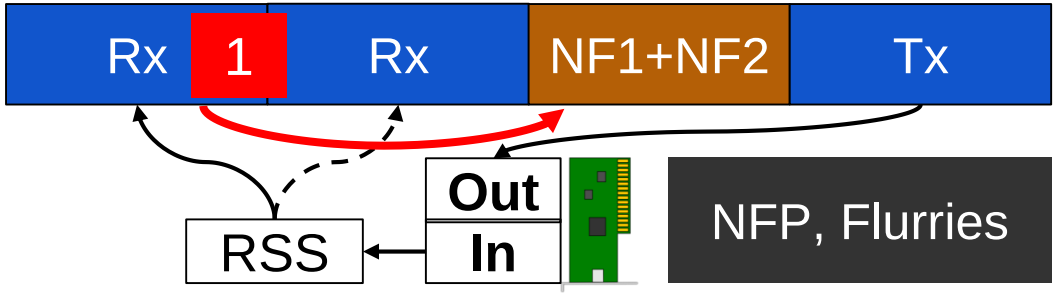


Software switch dispatching

4 Inter-Core Transfers

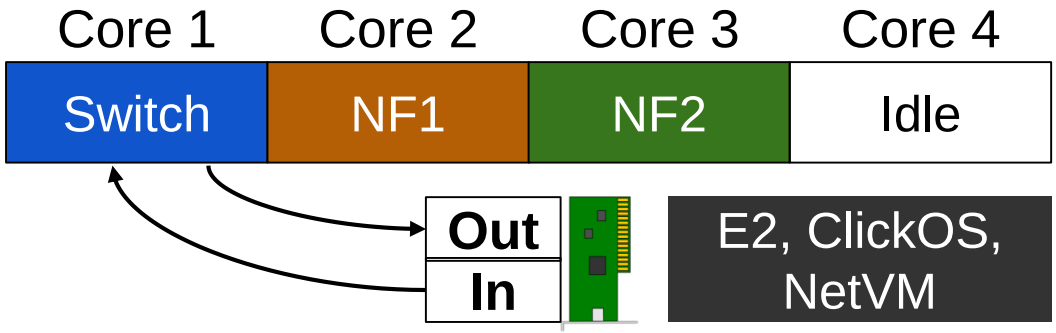



Pipeline dispatching with or without RSS

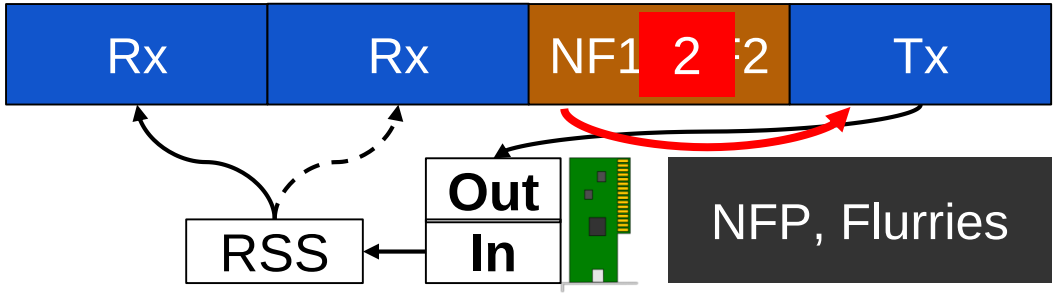


Software switch dispatching

4 Inter-Core Transfers

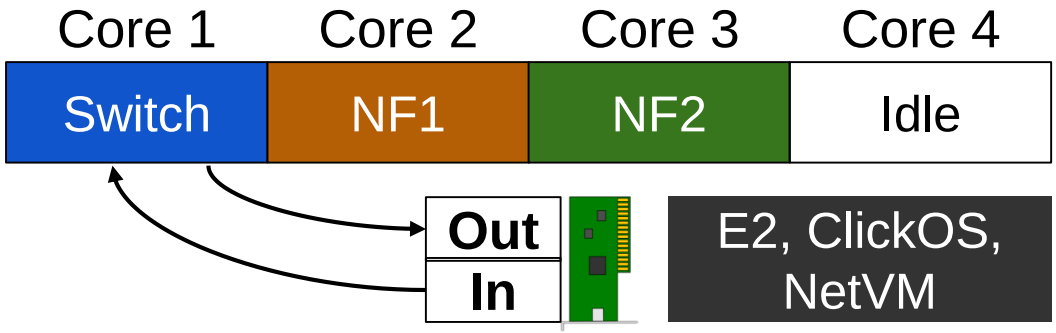



Pipeline dispatching with or without RSS

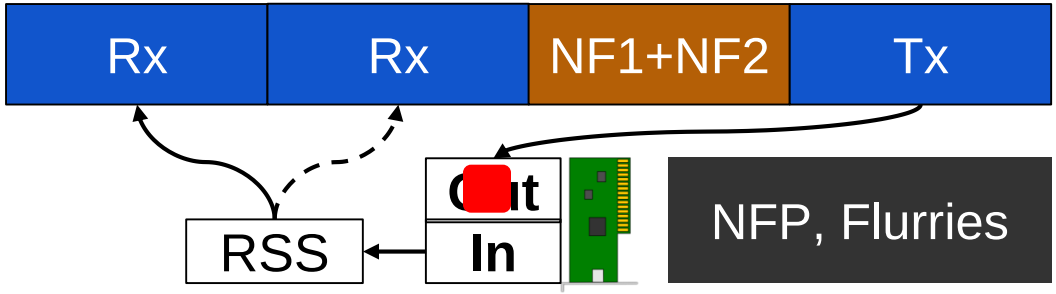


Software switch dispatching

4 Inter-Core Transfers

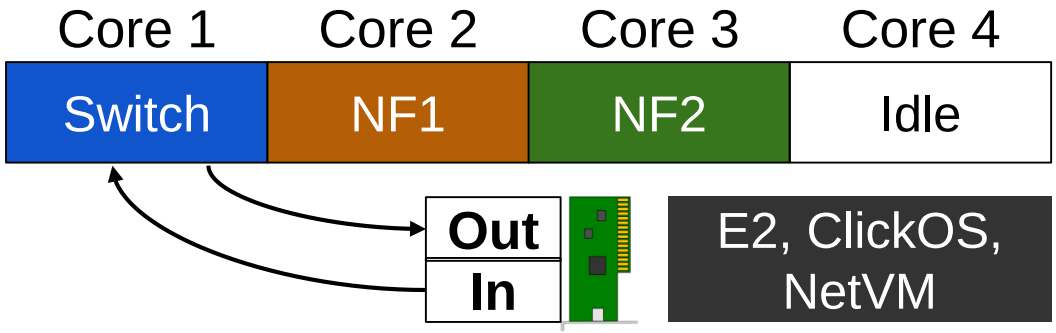



Pipeline dispatching with or without RSS



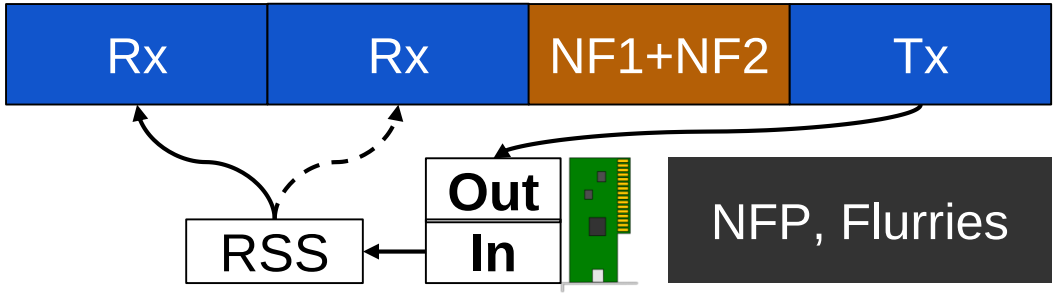
Software switch dispatching

4 Inter-Core Transfers



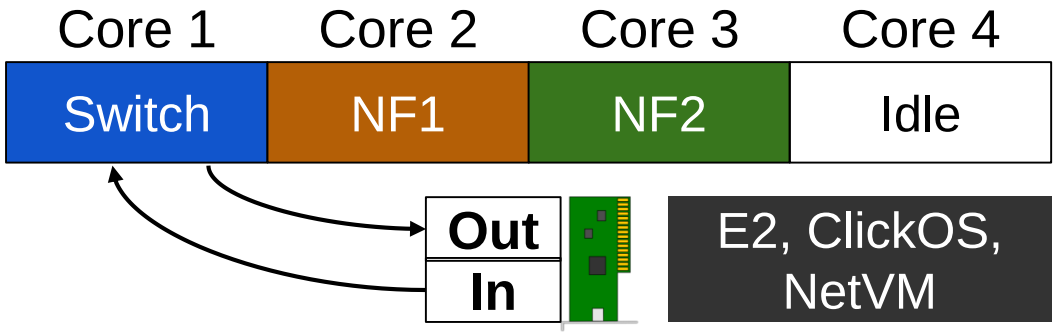

Pipeline dispatching with or without RSS

2 Inter-Core Transfers



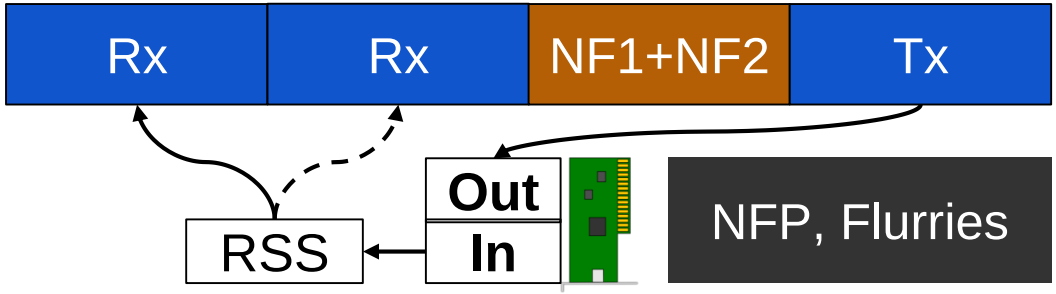
Software switch dispatching

4 Inter-Core Transfers

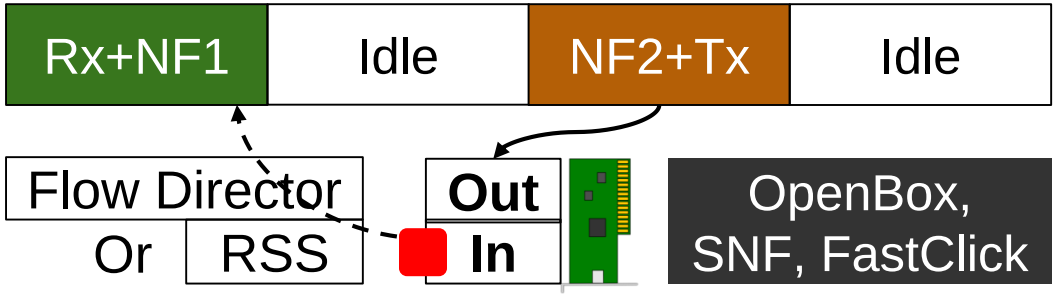



Pipeline dispatching with or without RSS

2 Inter-Core Transfers

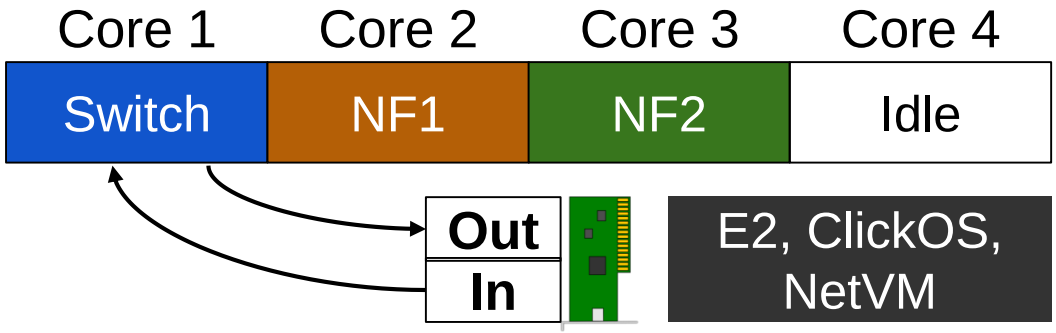



Rule or hash-based hardware dispatching



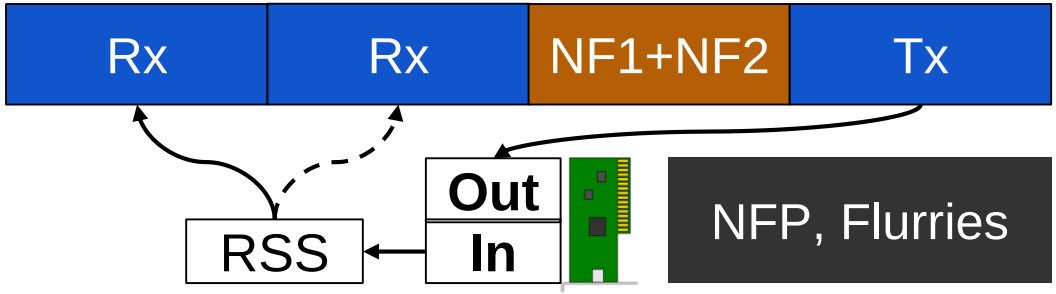
Software switch dispatching

**4 Inter-Core Transfers**

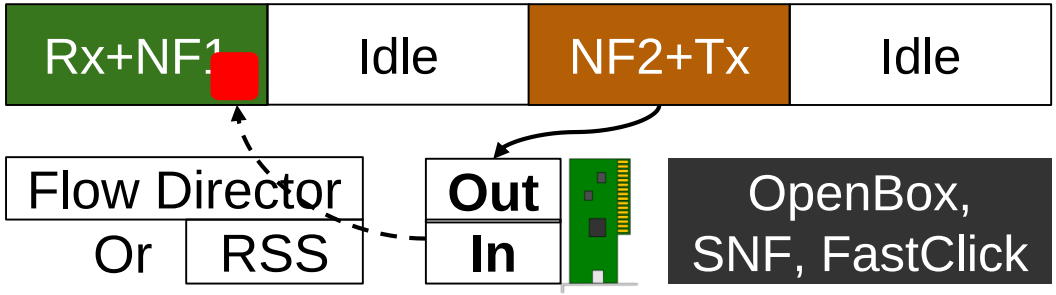



Pipeline dispatching with or without RSS

**2 Inter-Core Transfers**

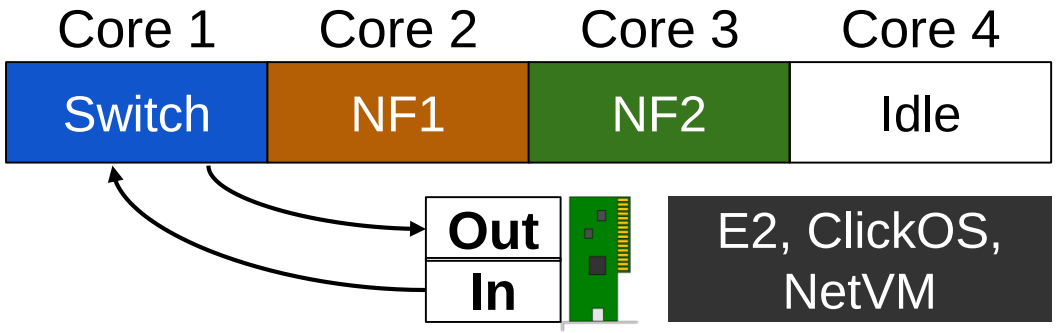



Rule or hash-based hardware dispatching



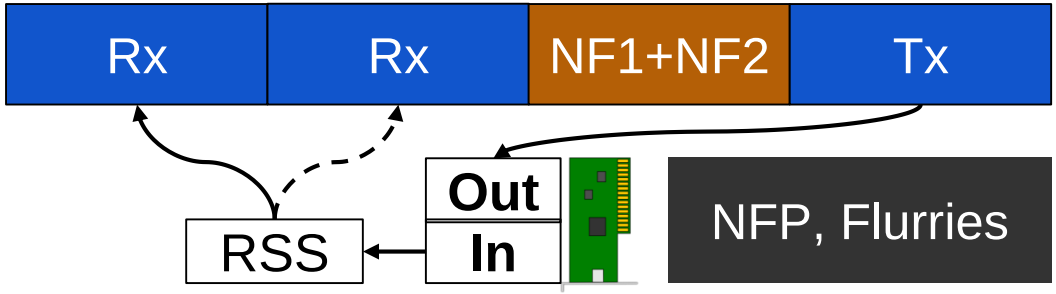
Software switch dispatching

4 Inter-Core Transfers

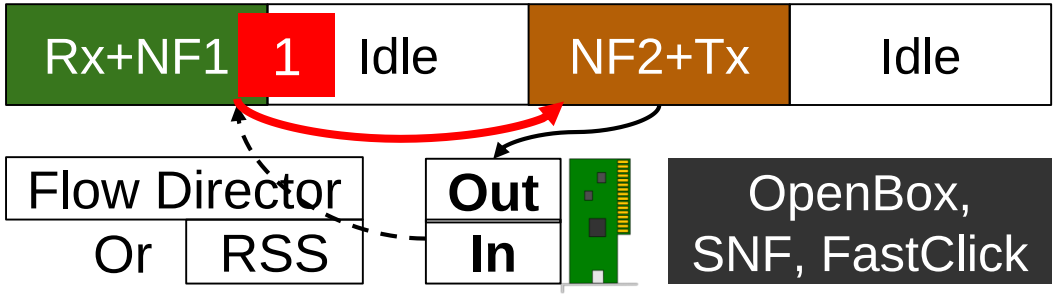



Pipeline dispatching with or without RSS

2 Inter-Core Transfers

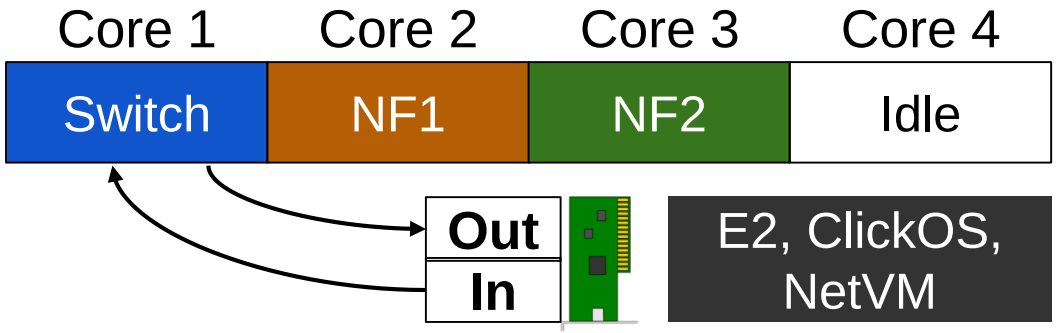



Rule or hash-based hardware dispatching



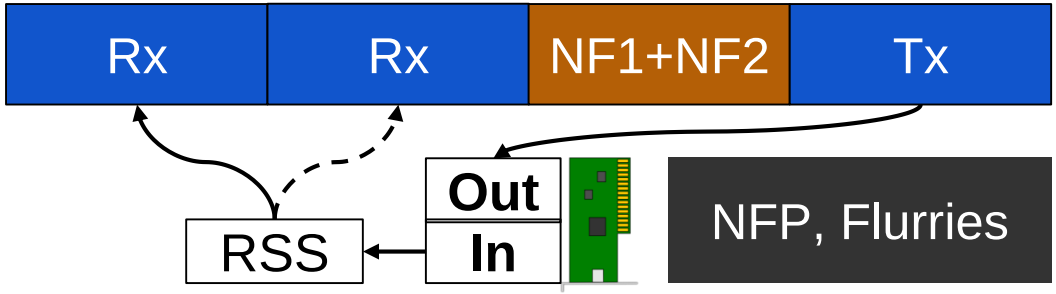
Software switch dispatching

**4 Inter-Core Transfers**

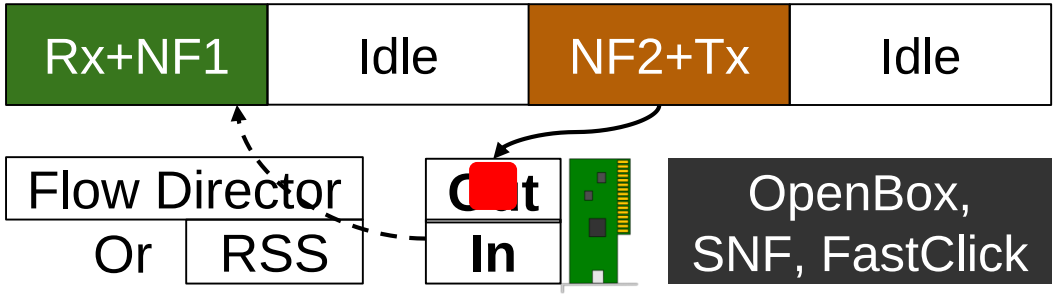



Pipeline dispatching with or without RSS

**2 Inter-Core Transfers**

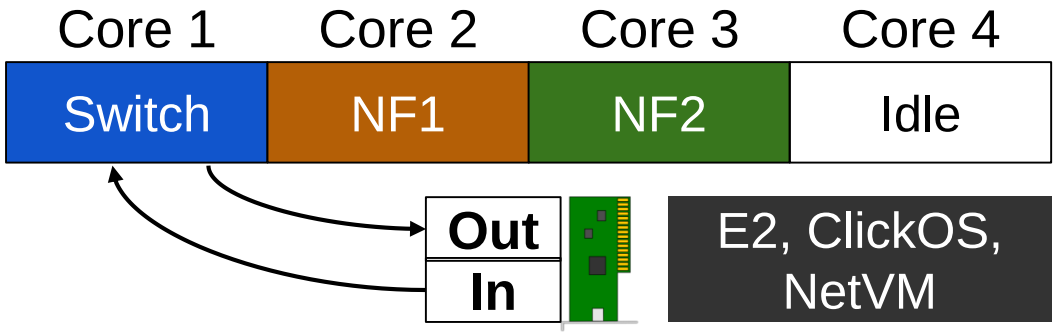



Rule or hash-based hardware dispatching



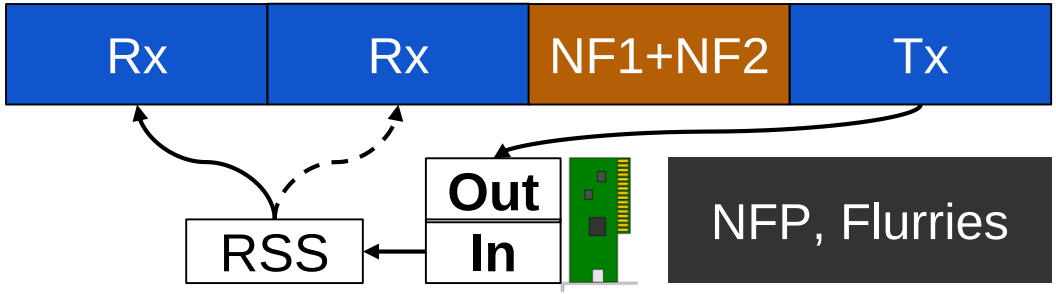
Software switch dispatching

**4 Inter-Core Transfers**



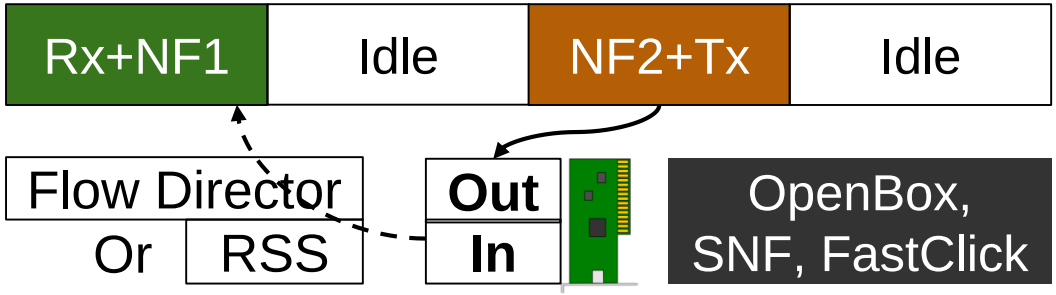

Pipeline dispatching with or without RSS

**2 Inter-Core Transfers**

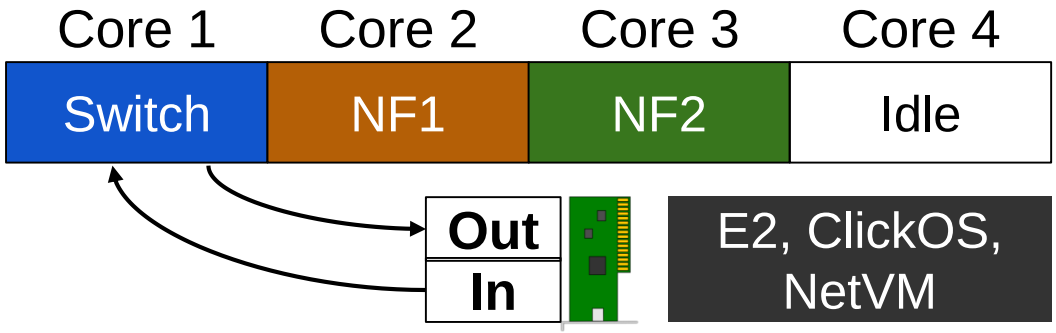



Rule or hash-based hardware dispatching

**1 Inter-Core Transfer**



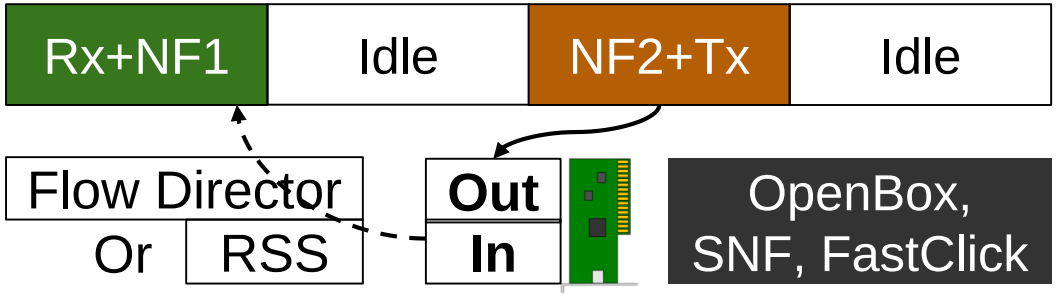
Software switch dispatching



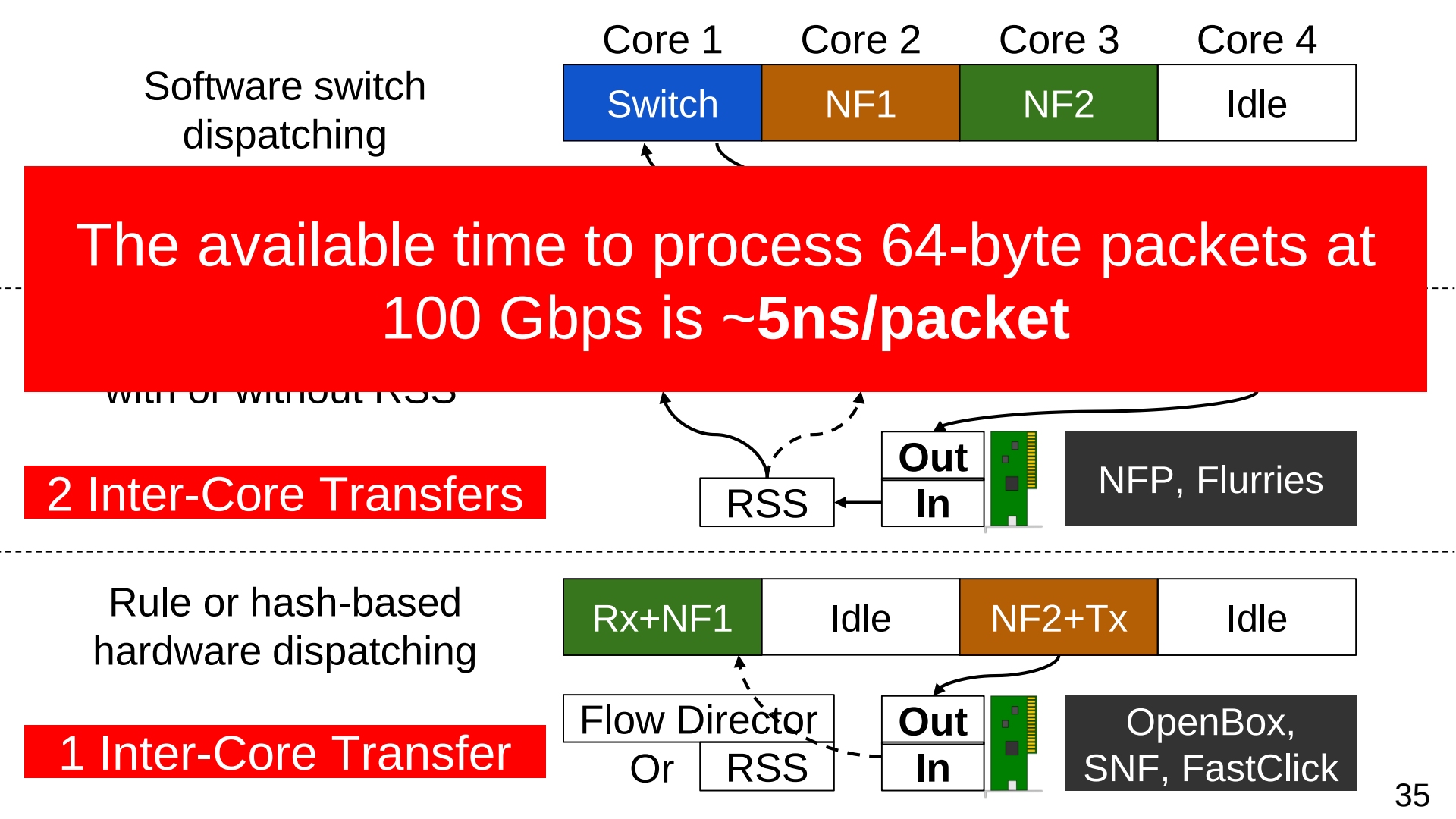

4 Inter-Core Transfers

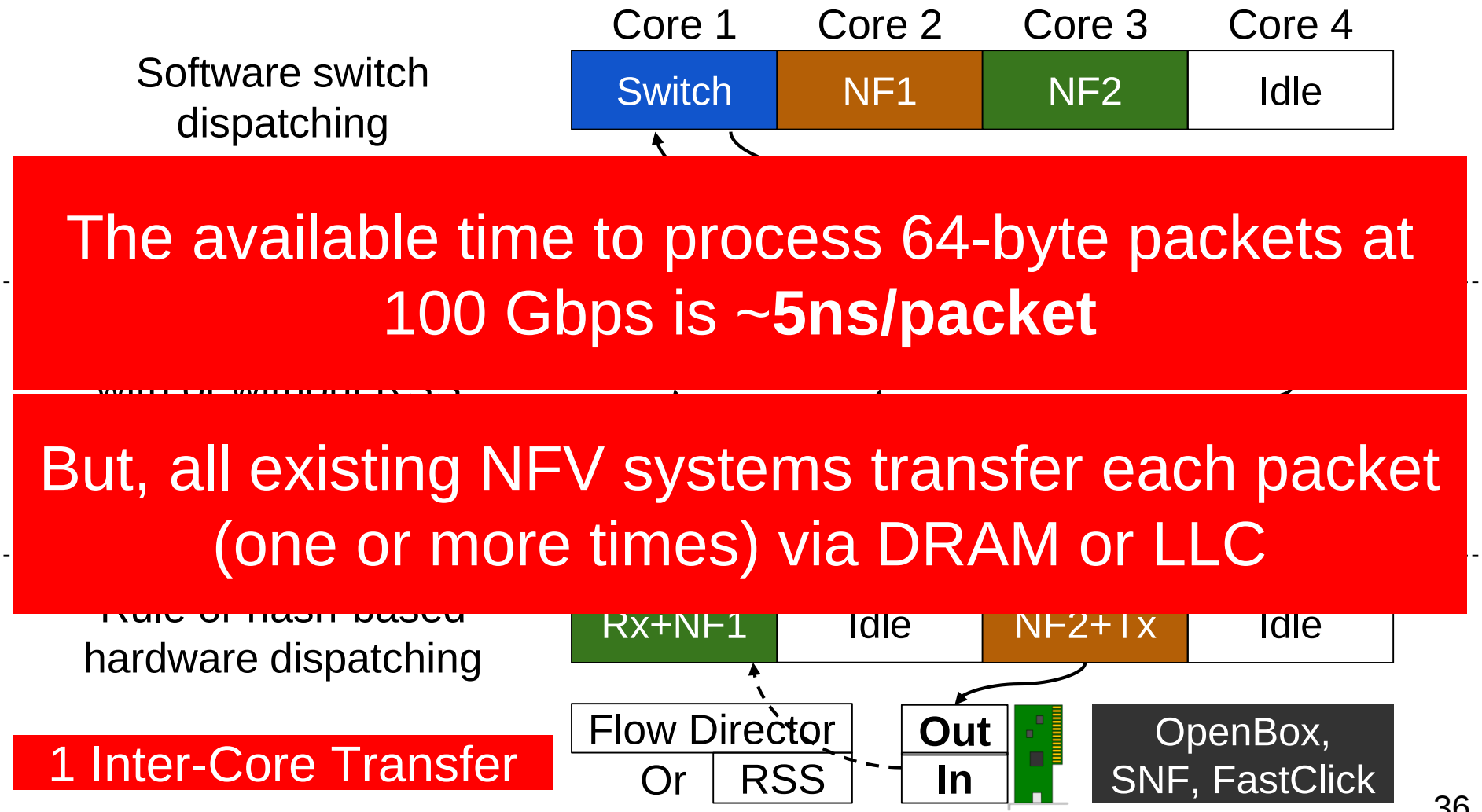
Fastest system → fewest inter-core transfers

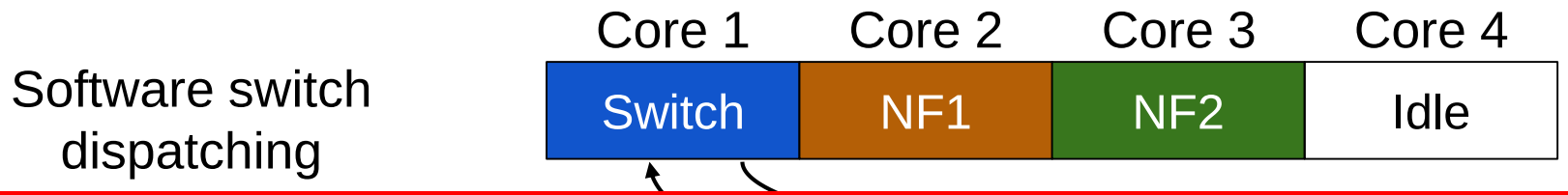
Rule or hash-based hardware dispatching




1 Inter-Core Transfer


Core 1
Core 2
Core 3
Core 4
Software switch dispatching
Switch
NF1
NF2
Idle
The available time to process 64-byte packets at 100 Gbps is ~5ns/packet
with or without RSS
2 Inter-Core Transfers
RSS
Out
In
NFP, Flurries
Rule or hash-based hardware dispatching
Rx+NF1
Idle
NF2+Tx
Idle
1 Inter-Core Transfer
Flow Director
Or
RSS
Out
In
OpenBox, SNF, FastClick


Core 1
Core 2
Core 3
Core 4
Software switch dispatching
Switch
NF1
NF2
Idle
The available time to process 64-byte packets at 100 Gbps is ~5ns/packet
But, all existing NFV systems transfer each packet (one or more times) via DRAM or LLC
Rule or hash based hardware dispatching
Rx+NF1
Idle
NF2+Tx
Idle
Flow Director
Or
RSS
Out
In
OpenBox, SNF, FastClick
1 Inter-Core Transfer




The available time to process 64-byte packets at 100 Gbps is ~**5ns/packet**

But, all existing NFV systems transfer each packet (one or more times) via DRAM or LLC

Such a transfer takes several nanoseconds itself!

# Zero Inter-core Transfers with Metron

## Problems

Greatly underutilized hardware

## Challenges

Hardware heterogeneity

## Solutions

Unified hardware management abstractions

# Zero Inter-core Transfers with Metron

## Problems

- Greatly underutilized hardware
- Too many inter-core transfers

## Challenges

- Hardware heterogeneity
- Dynamic adaptation

## Solutions

- Unified hardware management abstractions
- Dispatch traffic classes to CPU cores using tags. Map each tag to the correct core

# Zero Inter-core Transfers with Metron

## Problems

- Greatly underutilized hardware
- Too many inter-core transfers

## Challenges

- Hardware heterogeneity
- Dynamic adaptation

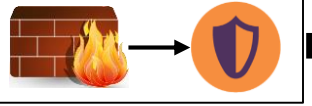
## Solutions

- Unified hardware management abstractions
- Dispatch traffic classes to CPU cores using tags. Map each tag to the correct core

## Results

- Up to 4.7x lower latency
- Up to 7.8x higher throughput
- 2.75-6.5x better efficiency

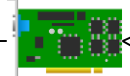
Firewall DPI



Input service chain to the controller

**Metron Controller**

Source

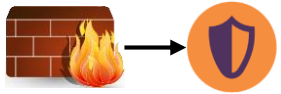


Core 1

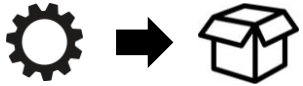
Core 2

**Server**

Firewall DPI



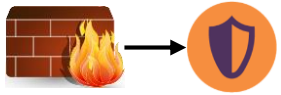
**Synthesized NF**



Input service chain to the controller

Katsikas et al.
SNF: synthesizing high performance NFV service chains
PeerJ Computer Science 2016

Firewall DPI



Input service chain to the controller

**Synthesized NF**

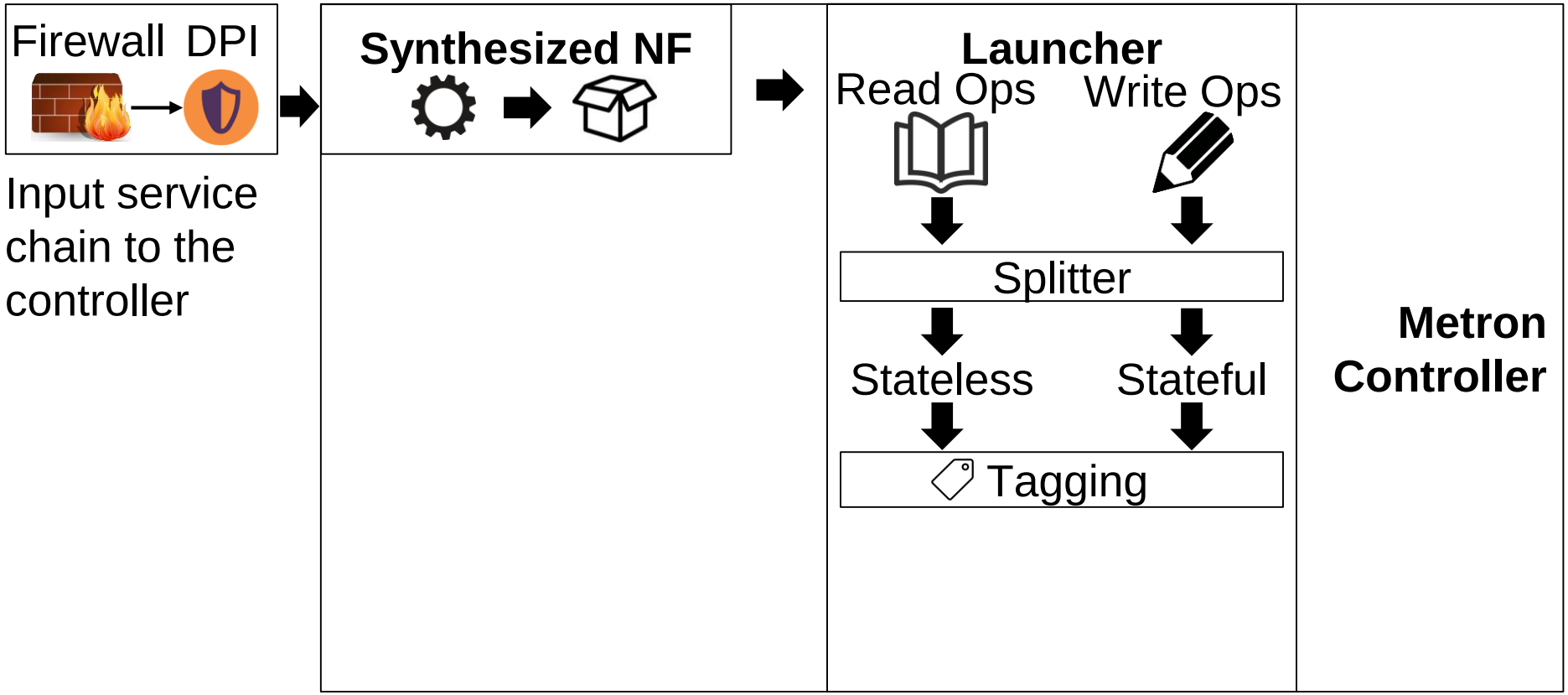
**Launcher**

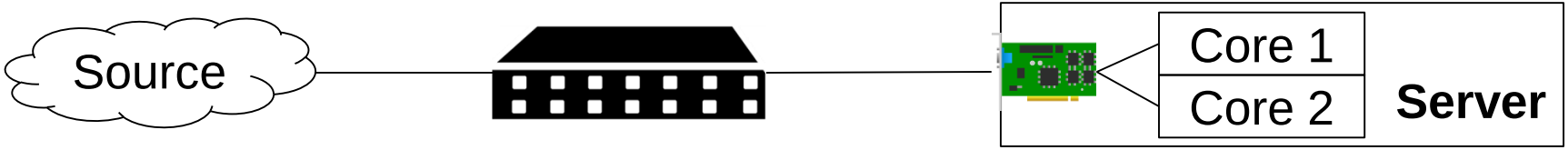
Read Ops Write Ops

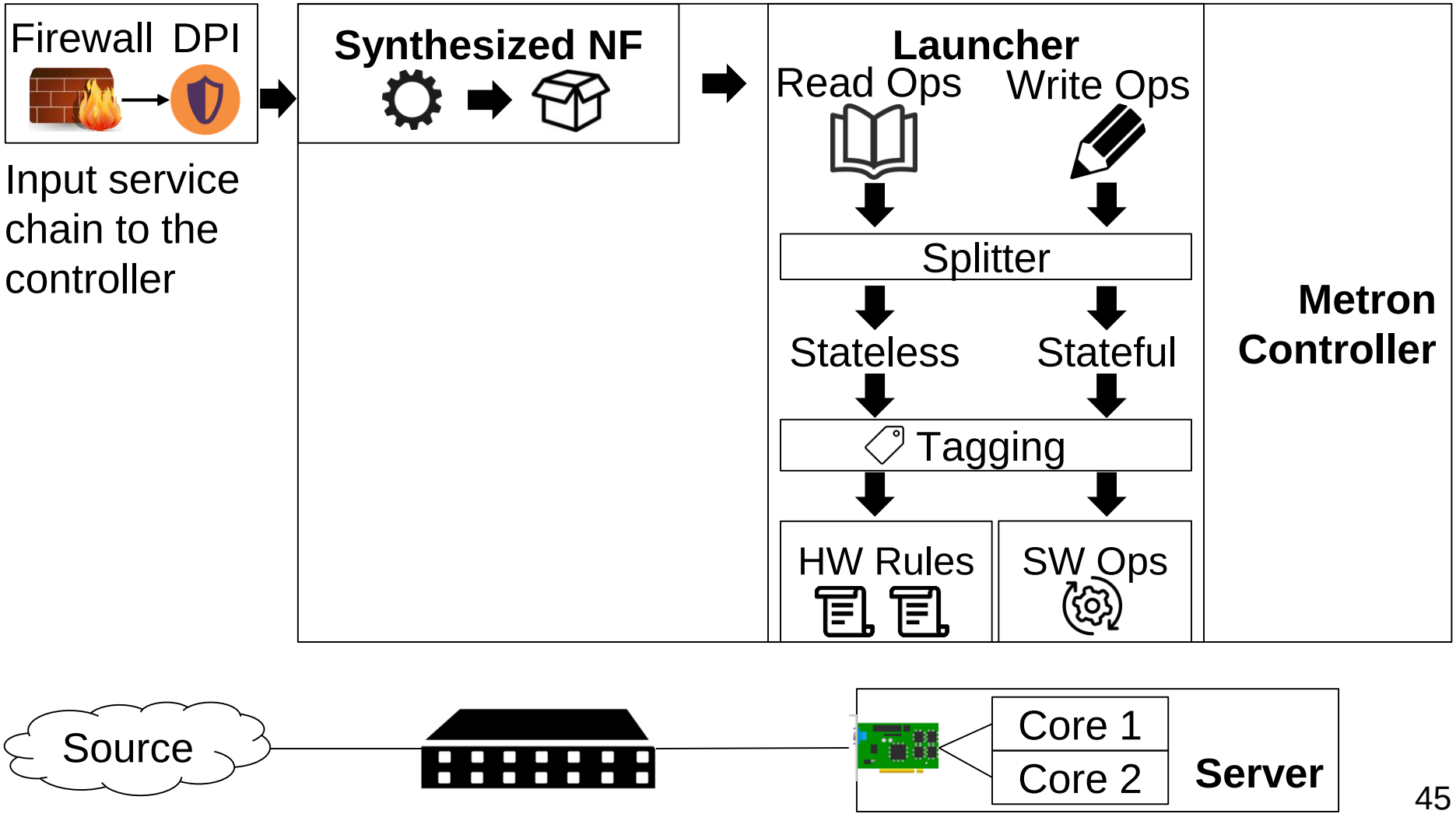
Splitter

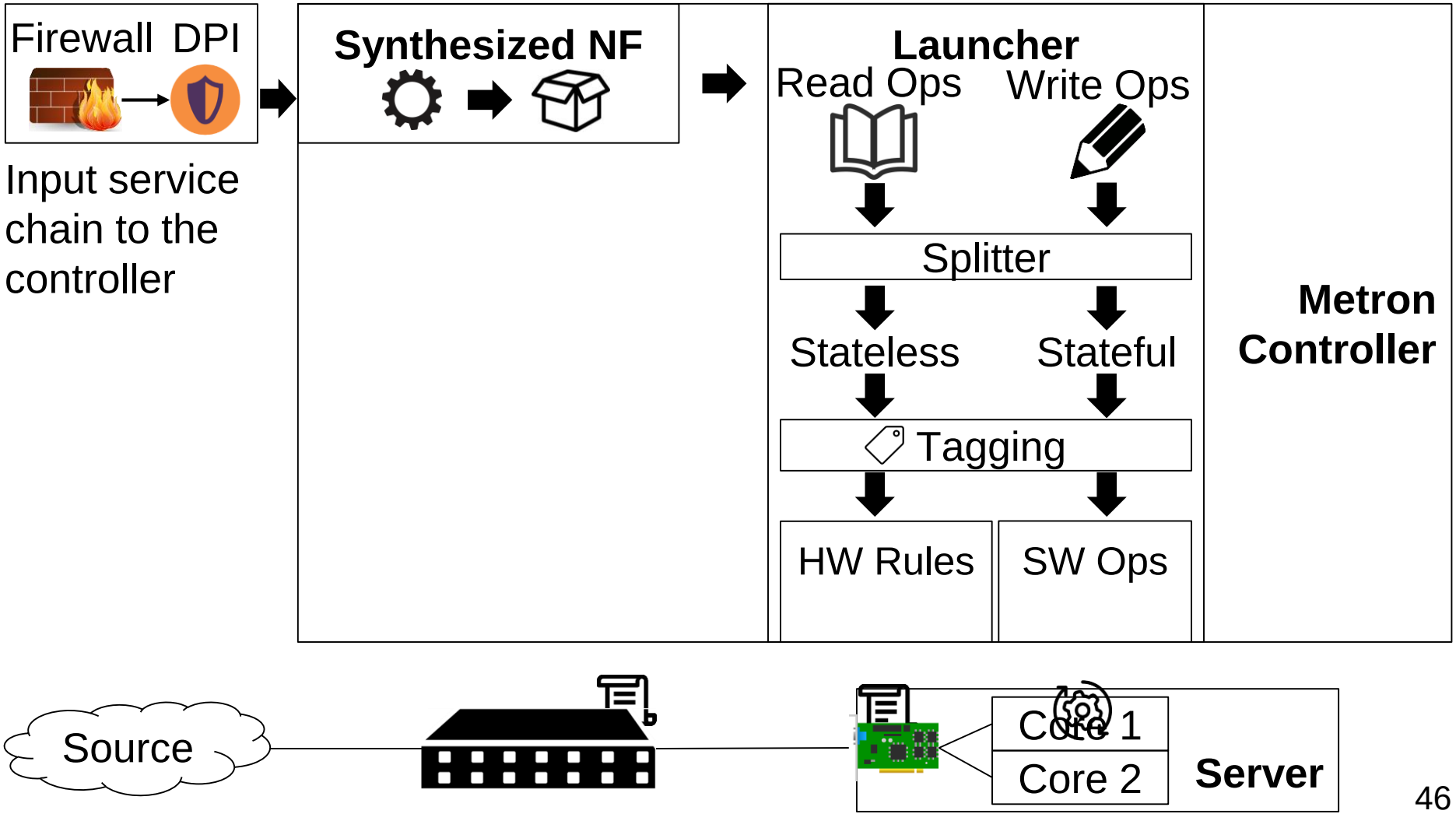
Stateless Stateful

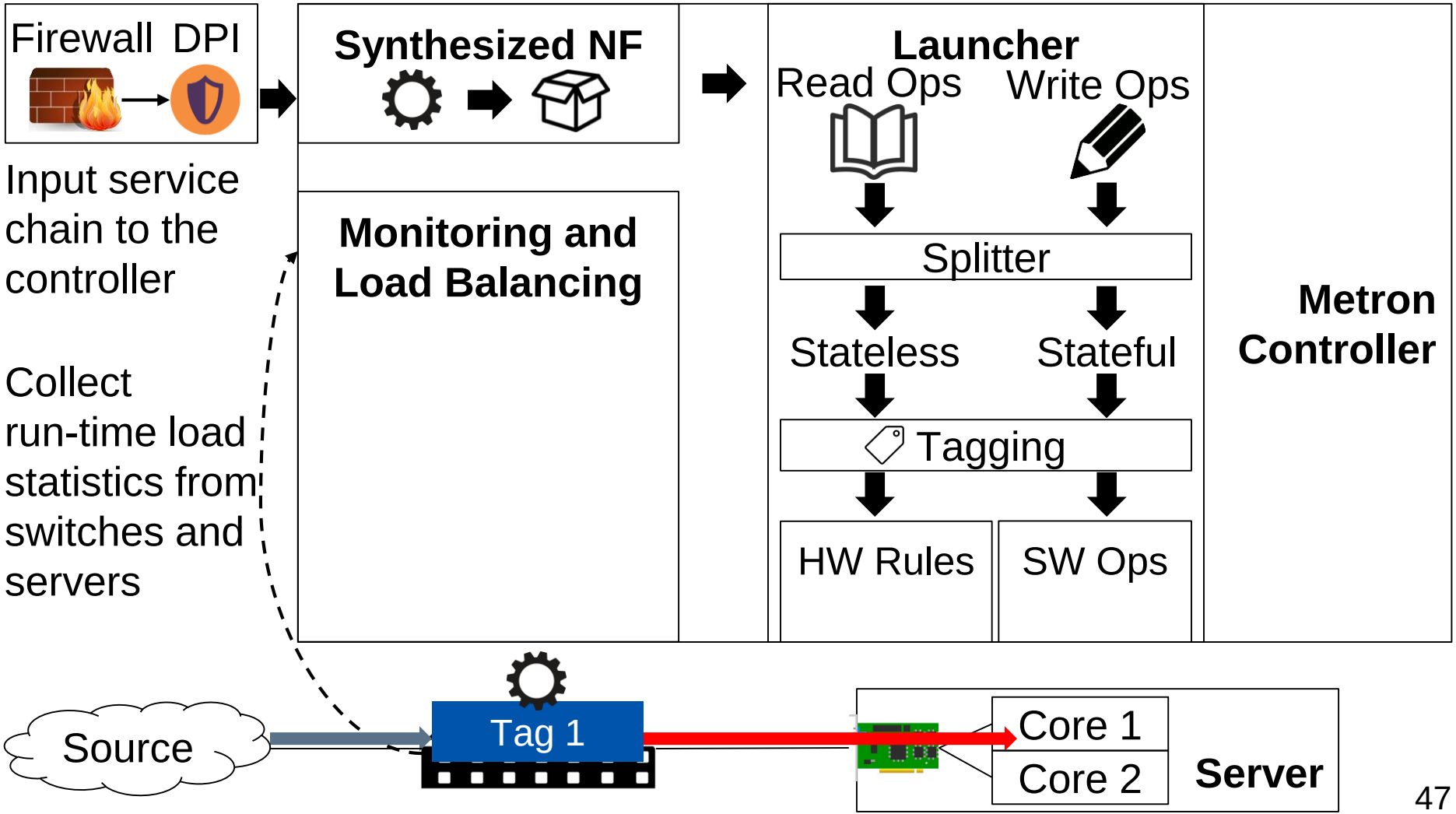
**Metron Controller**

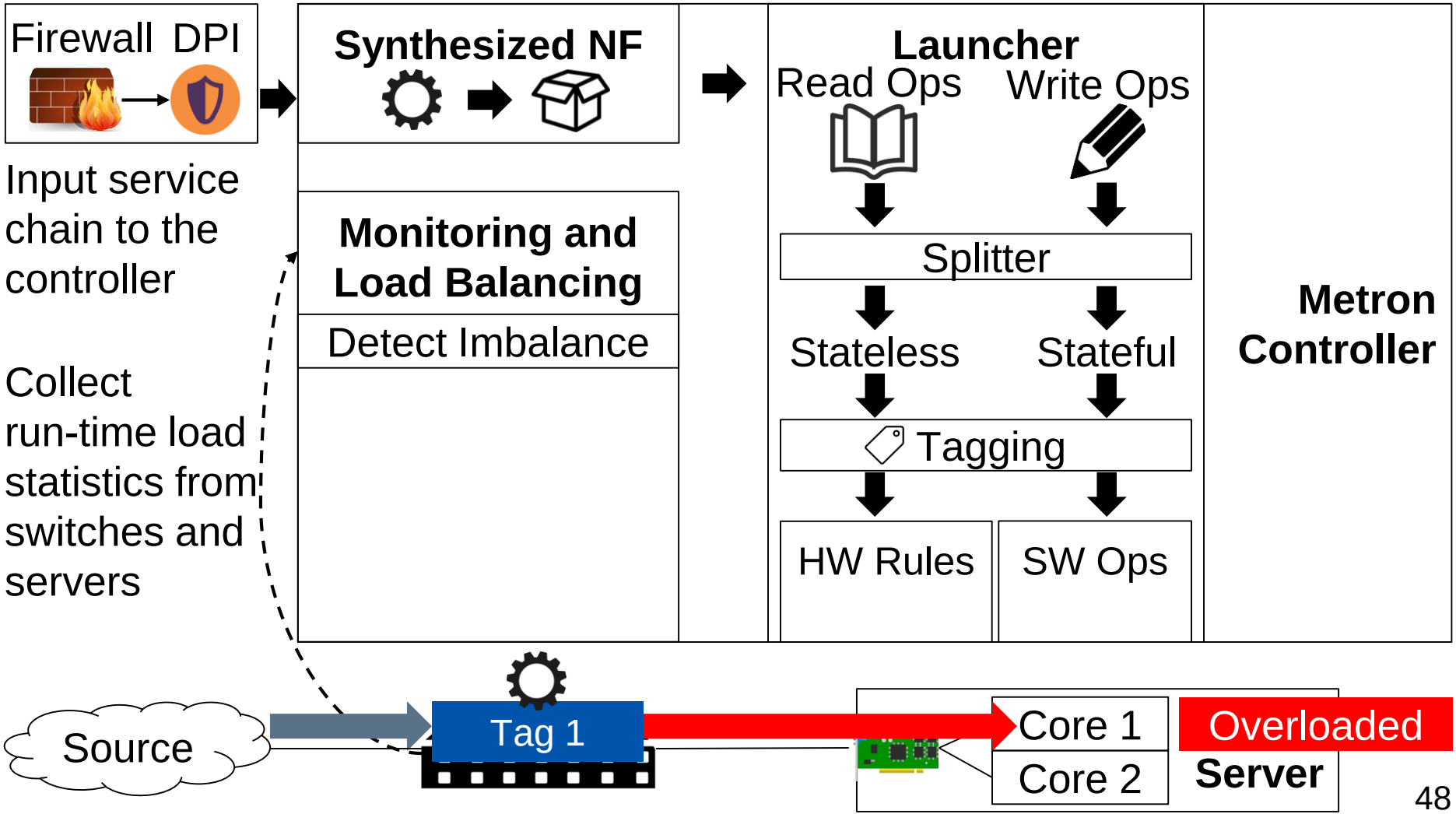

Firewall DPI
Input service chain to the controller
Synthesized NF
Launcher
Read Ops
Write Ops
Splitter
Stateless
Stateful
Tagging
Metron Controller

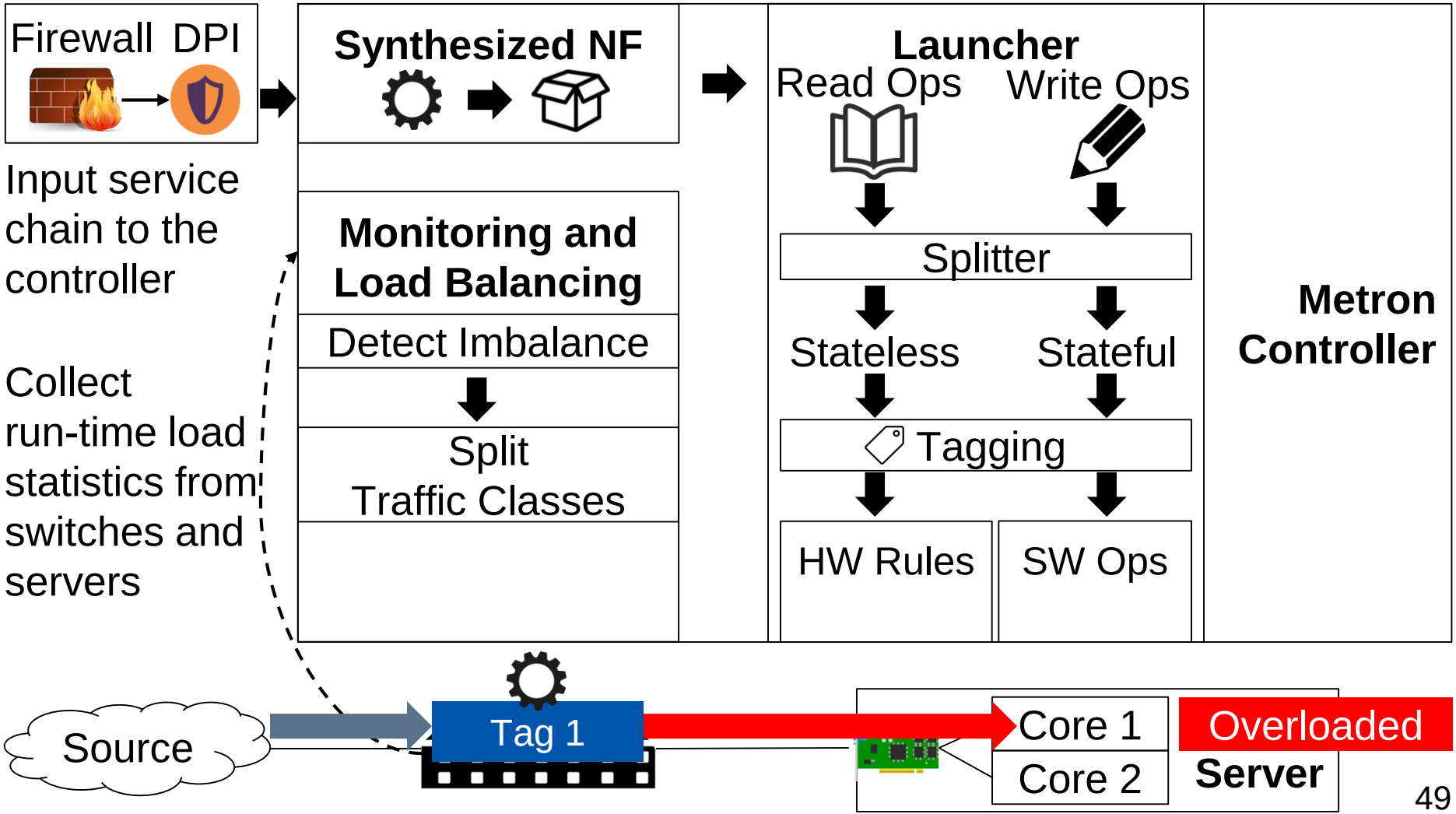

Source
Core 1
Core 2
Server

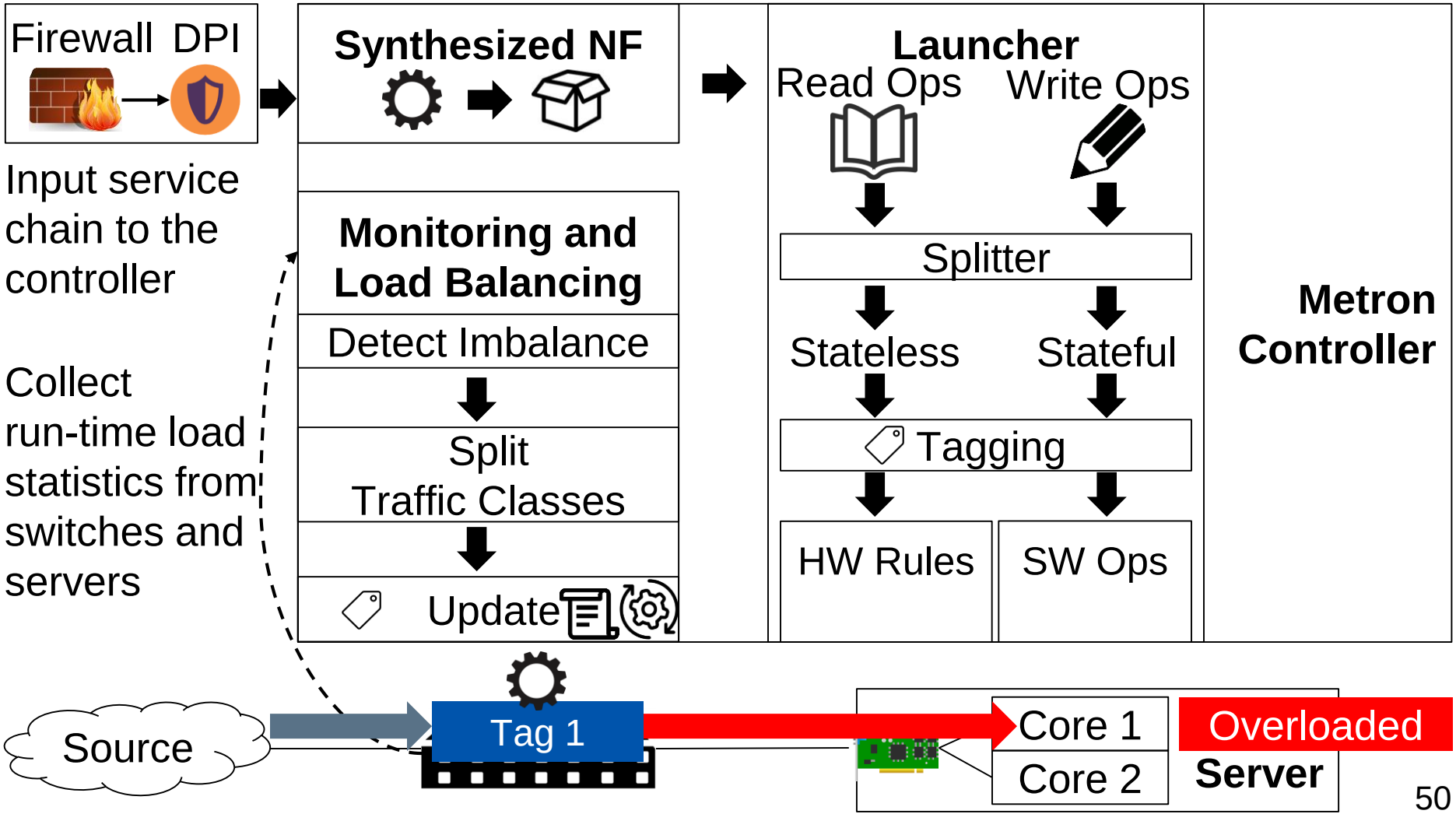

Firewall DPI
Input service chain to the controller
Synthesized NF
Launcher
Read Ops
Write Ops
Splitter
Stateless
Stateful
Tagging
HW Rules
SW Ops
Metron Controller
Source
Core 1
Core 2
Server

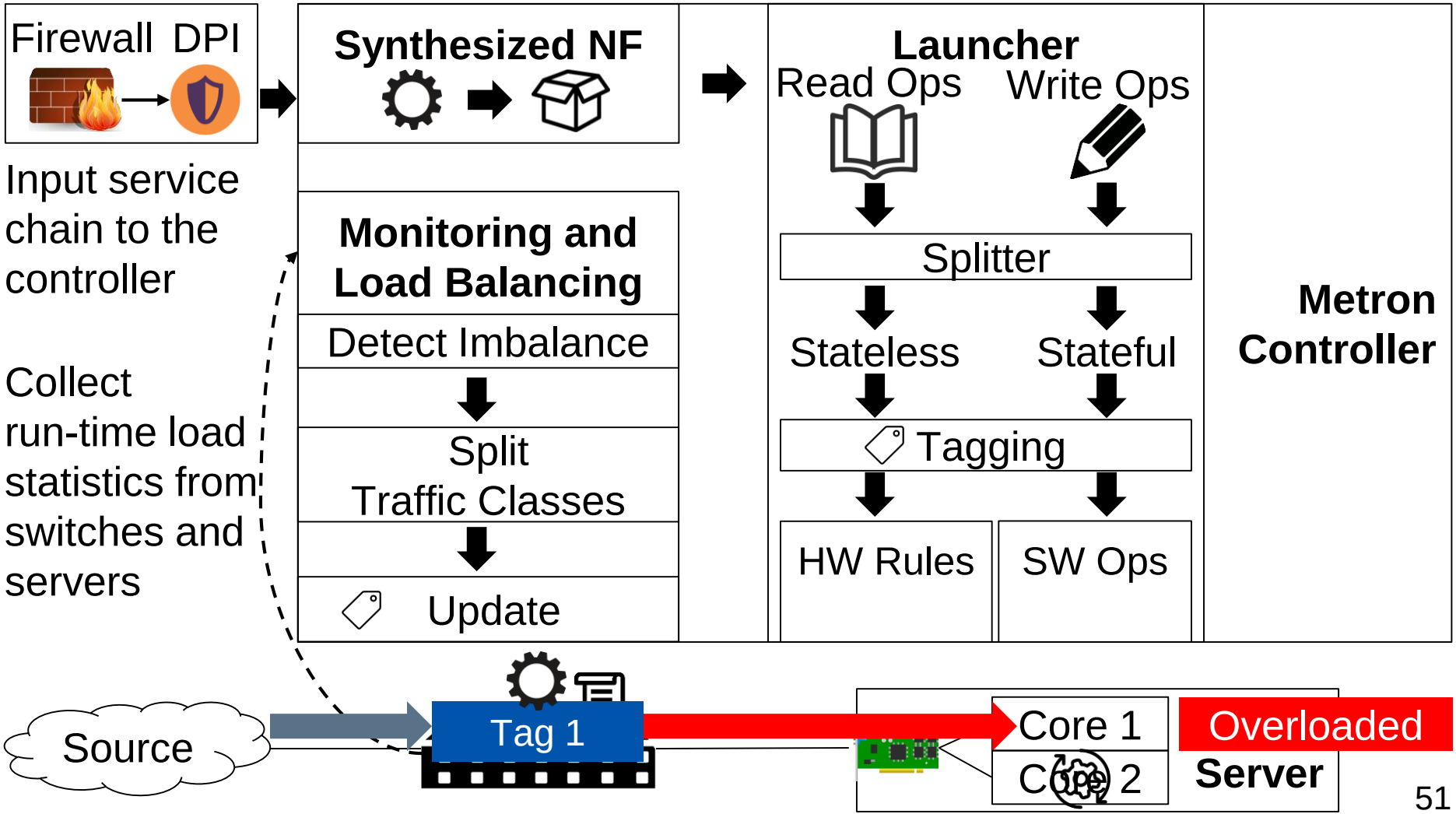

Firewall DPI
Input service chain to the controller
Synthesized NF
Launcher
Read Ops
Write Ops
Splitter
Stateless
Stateful
Tagging
HW Rules
SW Ops
Metron Controller
Source
Core 1
Core 2
Server

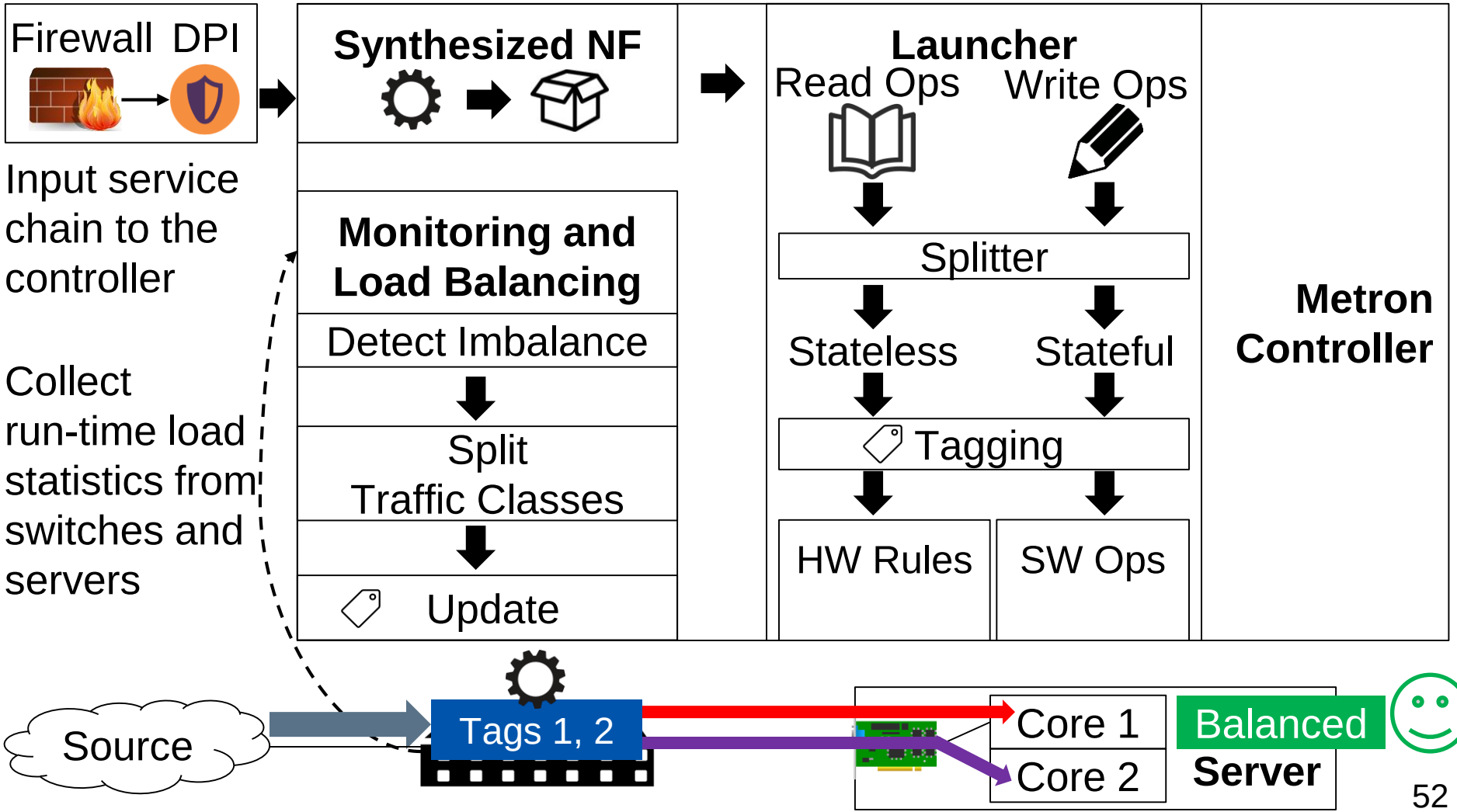

Firewall DPI
Synthesized NF
Launcher
Read Ops
Write Ops
Splitter
Stateless
Stateful
Tagging
HW Rules
SW Ops
Metron Controller
Monitoring and Load Balancing
Input service chain to the controller
Collect run-time load statistics from switches and servers
Source
Tag 1
Core 1
Core 2
Server


Firewall DPI
Synthesized NF
Launcher
Read Ops
Write Ops
Splitter
Stateless
Stateful
Tagging
HW Rules
SW Ops
Metron Controller
Input service chain to the controller
Monitoring and Load Balancing
Detect Imbalance
Collect run-time load statistics from switches and servers
Source
Tag 1
Core 1
Core 2
Overloaded
Server

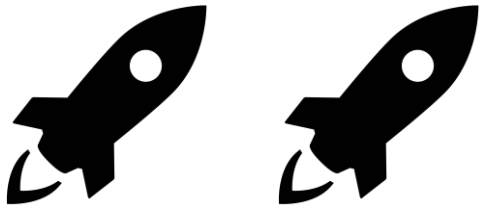

Firewall DPI
Synthesized NF
Launcher
Read Ops
Write Ops
Input service chain to the controller
Monitoring and Load Balancing
Detect Imbalance
Split Traffic Classes
Splitter
Stateless
Stateful
Tagging
HW Rules
SW Ops
Metron Controller
Collect run-time load statistics from switches and servers
Source
Tag 1
Core 1
Core 2
Overloaded
Server


Firewall DPI
Synthesized NF
Launcher
Read Ops
Write Ops
Input service chain to the controller
Monitoring and Load Balancing
Detect Imbalance
Split Traffic Classes
Update
Splitter
Stateless
Stateful
Tagging
HW Rules
SW Ops
Metron Controller
Collect run-time load statistics from switches and servers
Source
Tag 1
Core 1
Core 2
Overloaded
Server

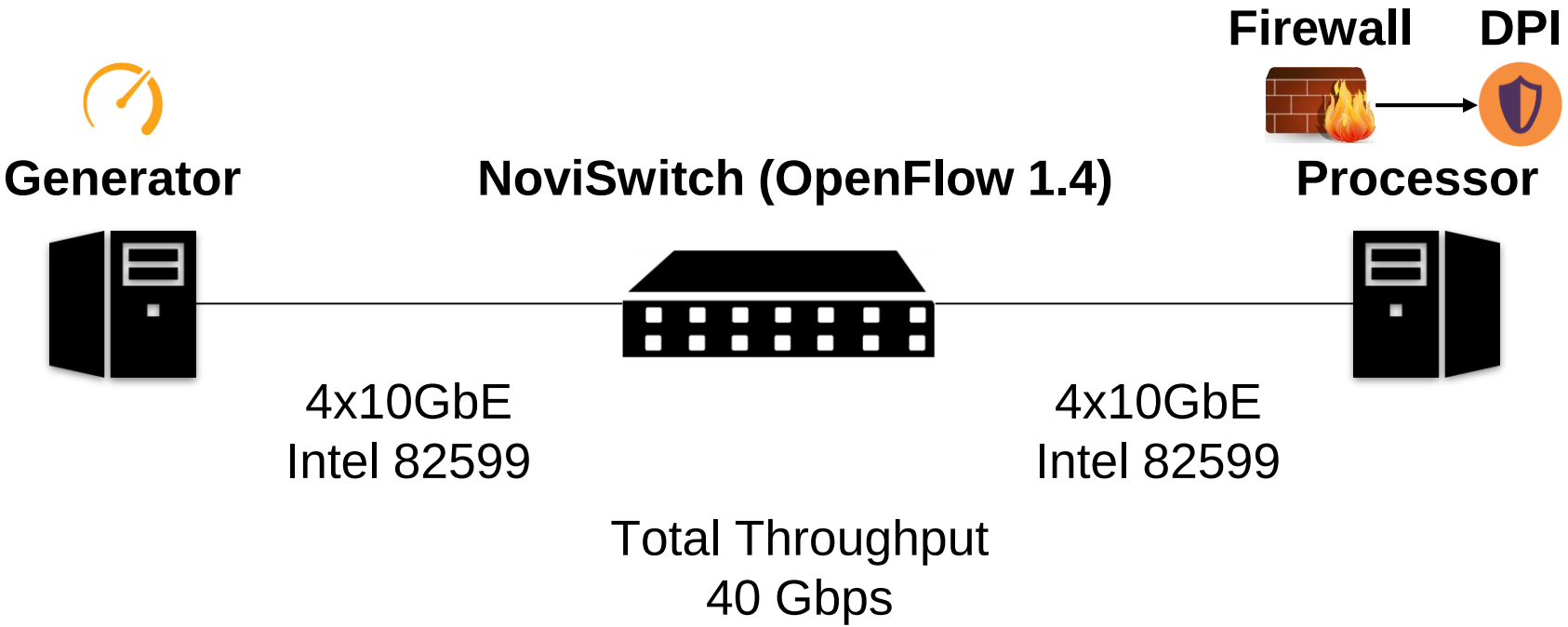

Firewall DPI
Synthesized NF
Launcher
Read Ops
Write Ops
Input service chain to the controller
Monitoring and Load Balancing
Detect Imbalance
Split Traffic Classes
Update
Splitter
Stateless
Stateful
Tagging
HW Rules
SW Ops
Metron Controller
Collect run-time load statistics from switches and servers
Source
Tag 1
Core 1
Core 2
Overloaded
Server


Firewall DPI
Synthesized NF
Launcher
Read Ops
Write Ops
Input service chain to the controller
Monitoring and Load Balancing
Detect Imbalance
Split Traffic Classes
Update
Splitter
Stateless
Stateful
Tagging
HW Rules
SW Ops
Metron Controller
Collect run-time load statistics from switches and servers
Source
Tags 1, 2
Core 1
Core 2
Balanced
Server

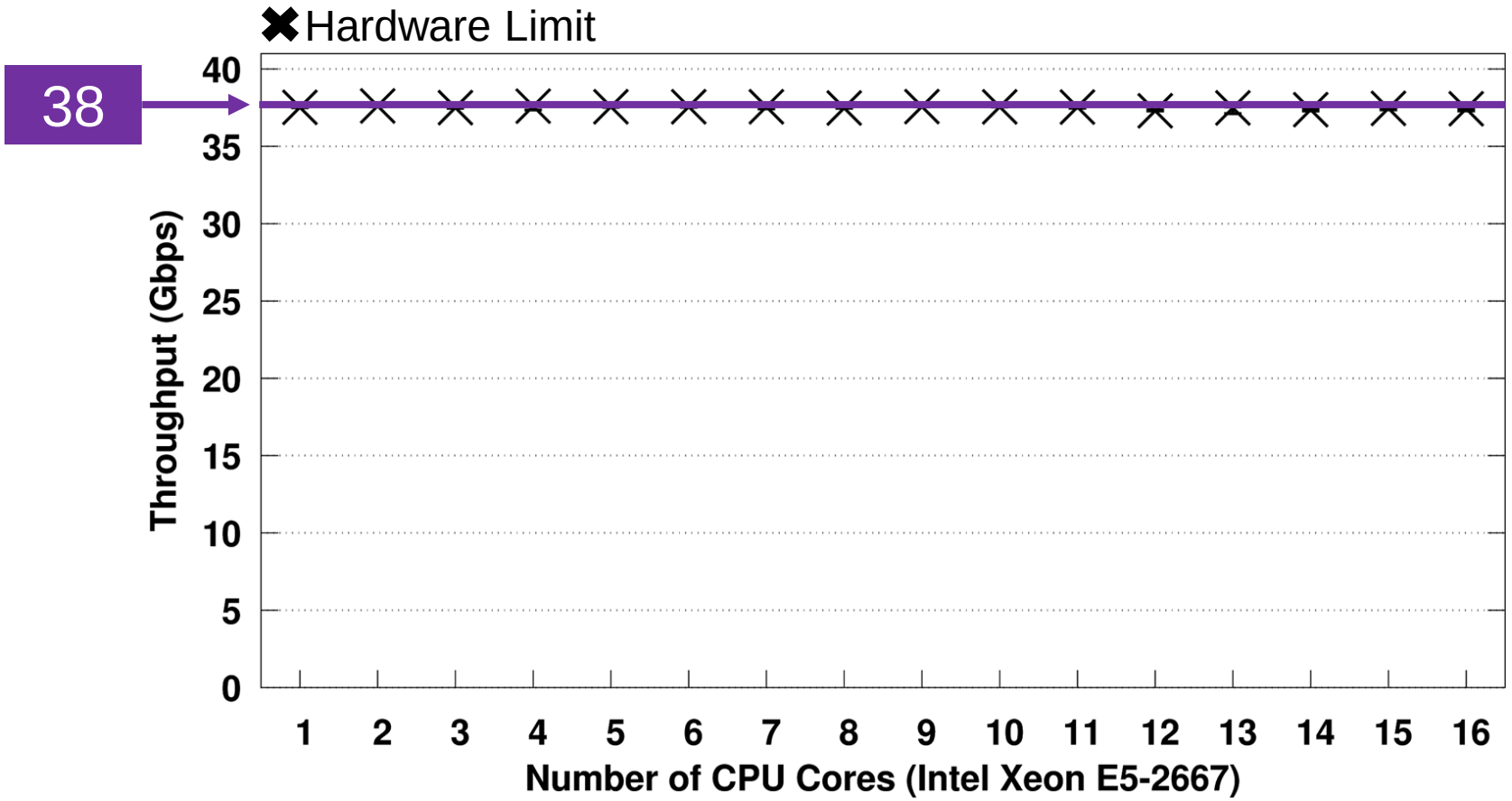
# Performance Evaluation



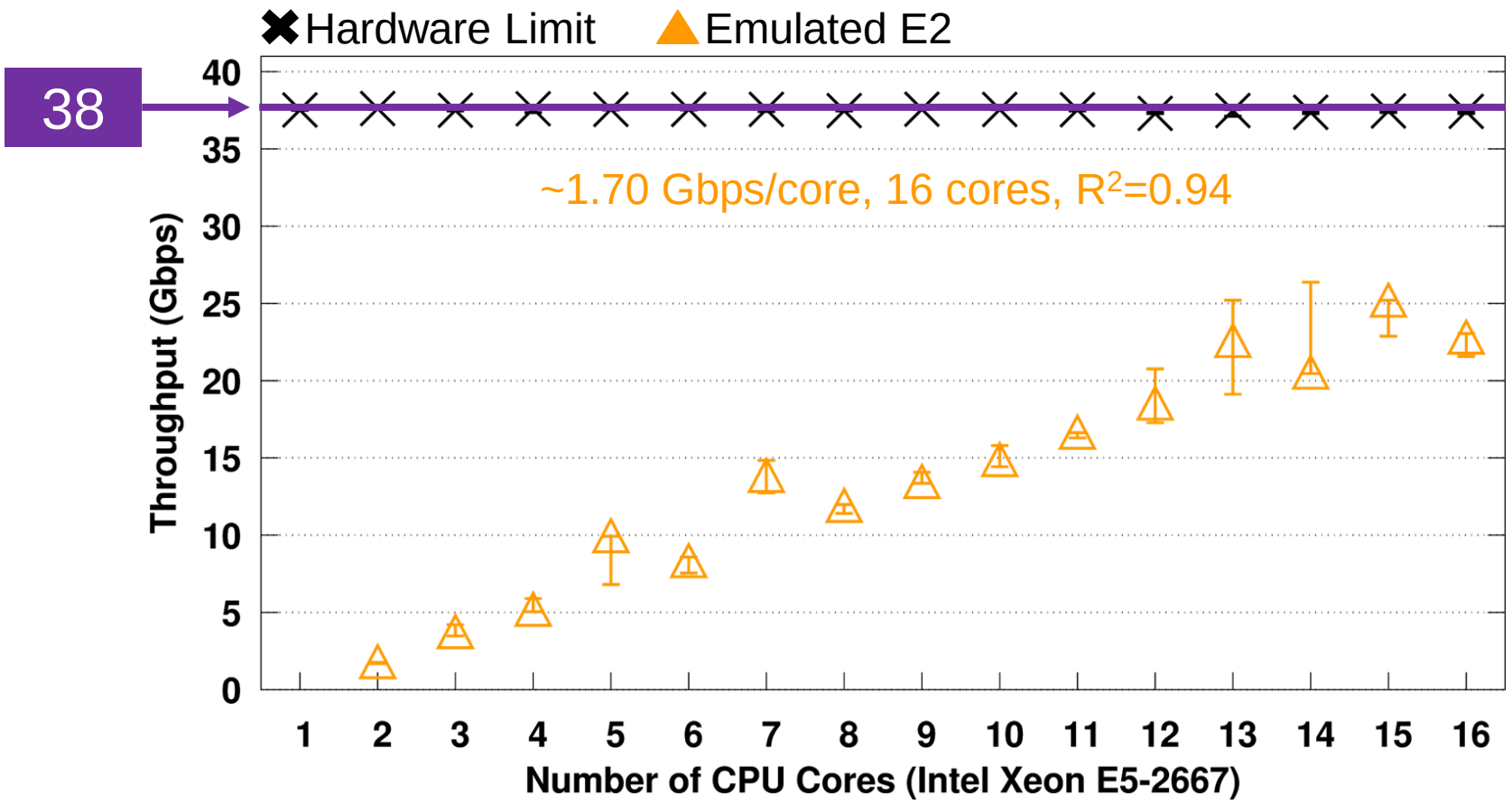
# Deployment at 40 Gbps



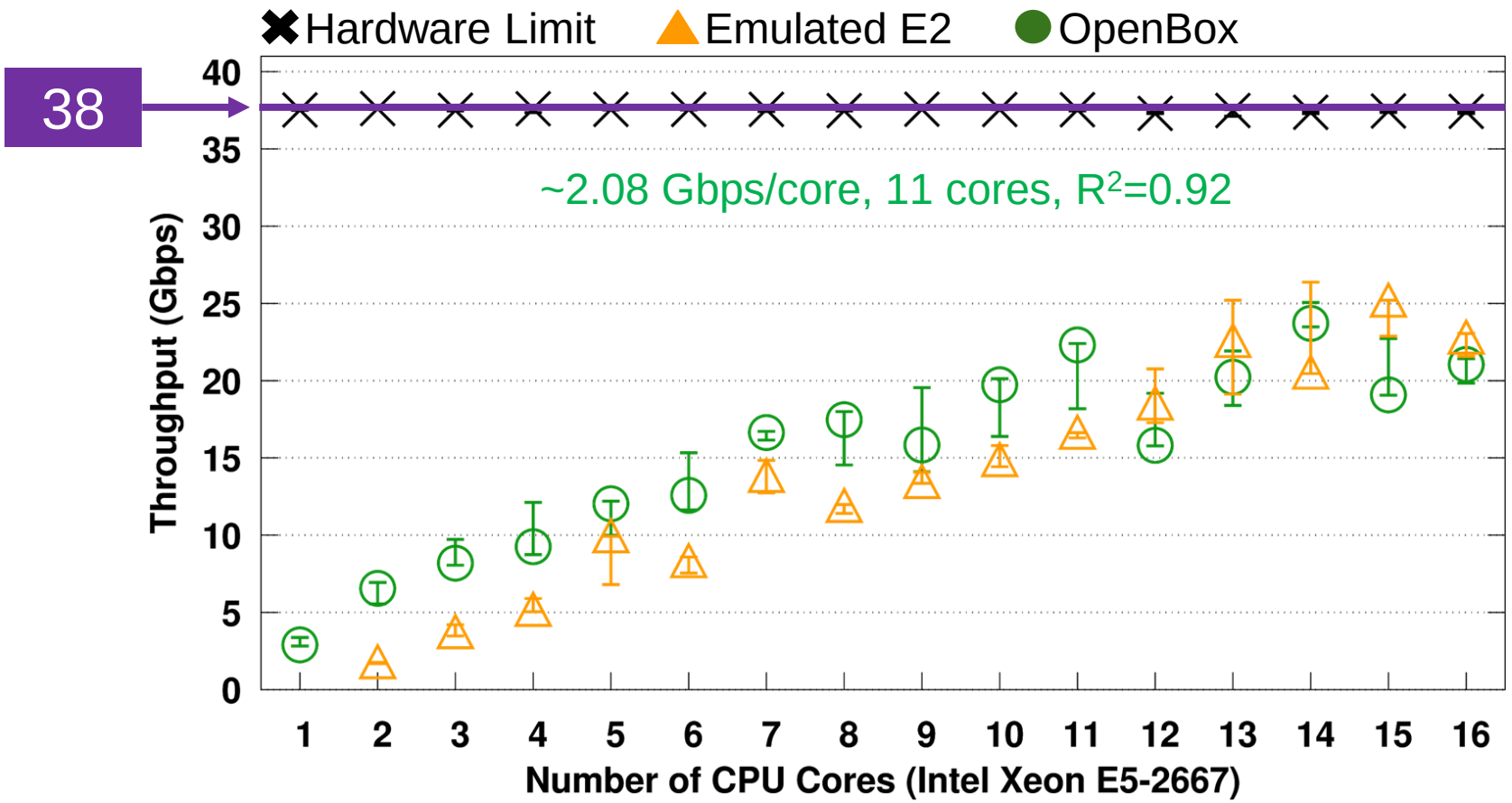
# Throughput at 40 Gbps



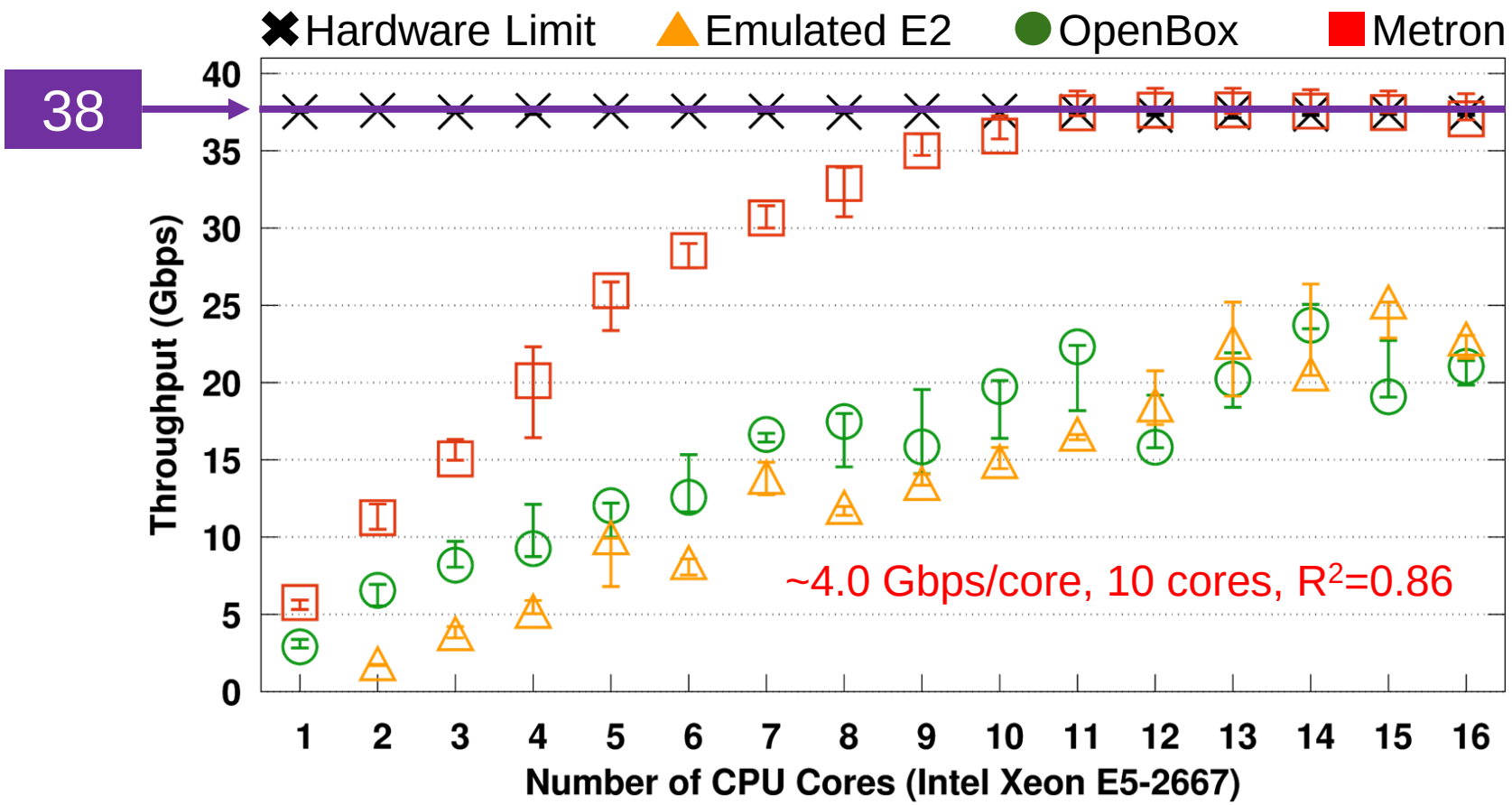
# Throughput at 40 Gbps



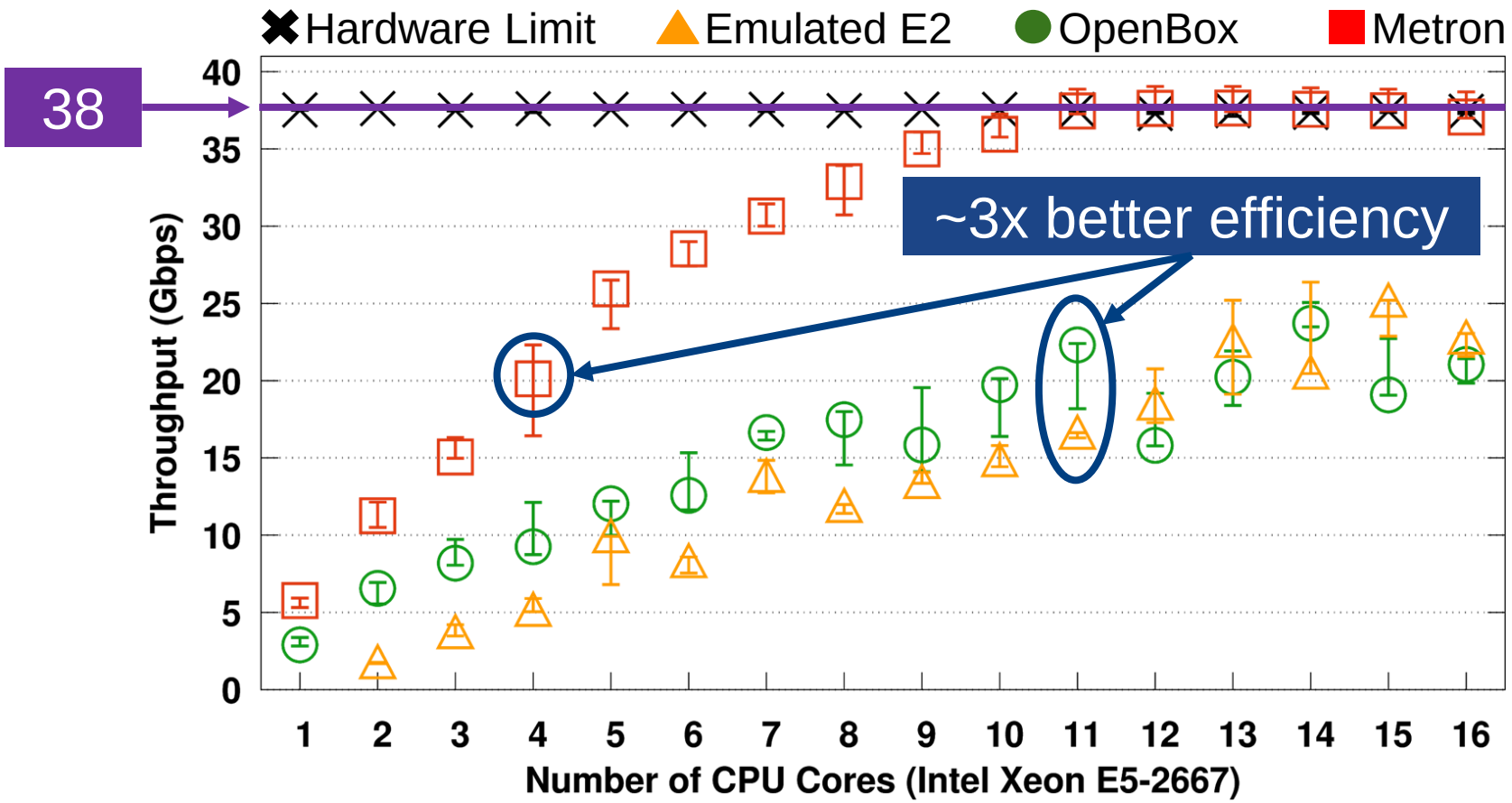
# Throughput at 40 Gbps



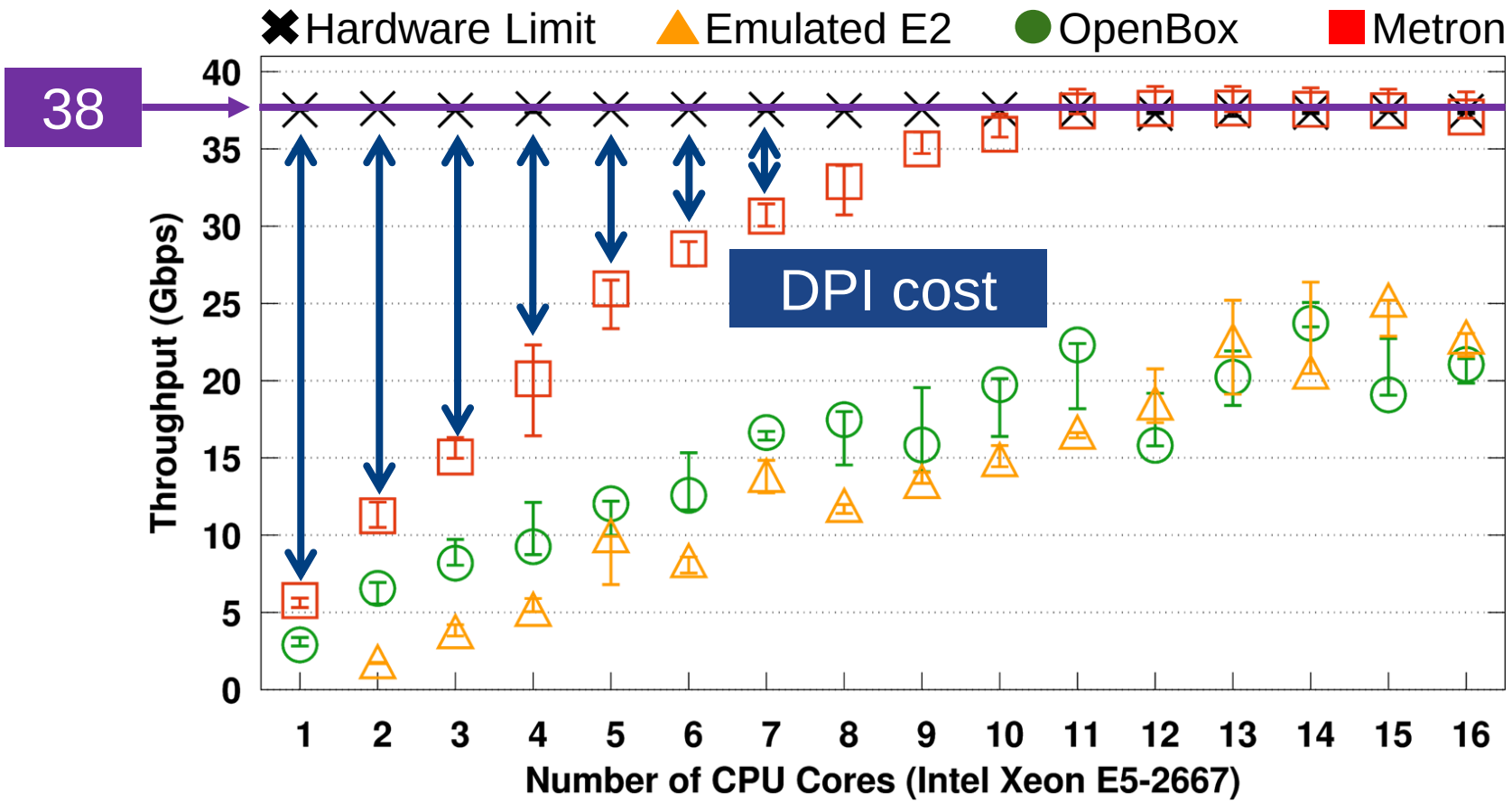
# Throughput at 40 Gbps



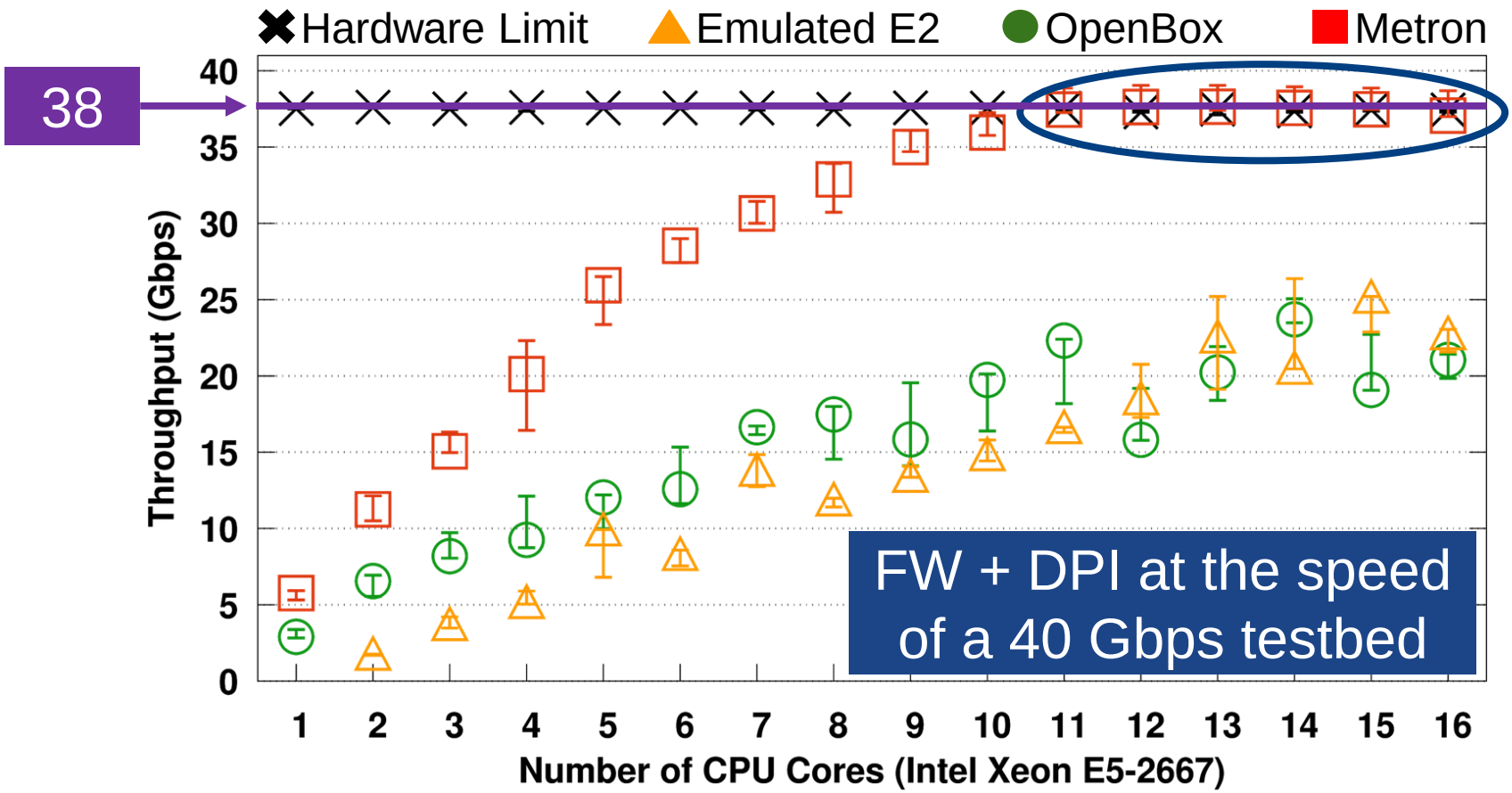
# Throughput at 40 Gbps



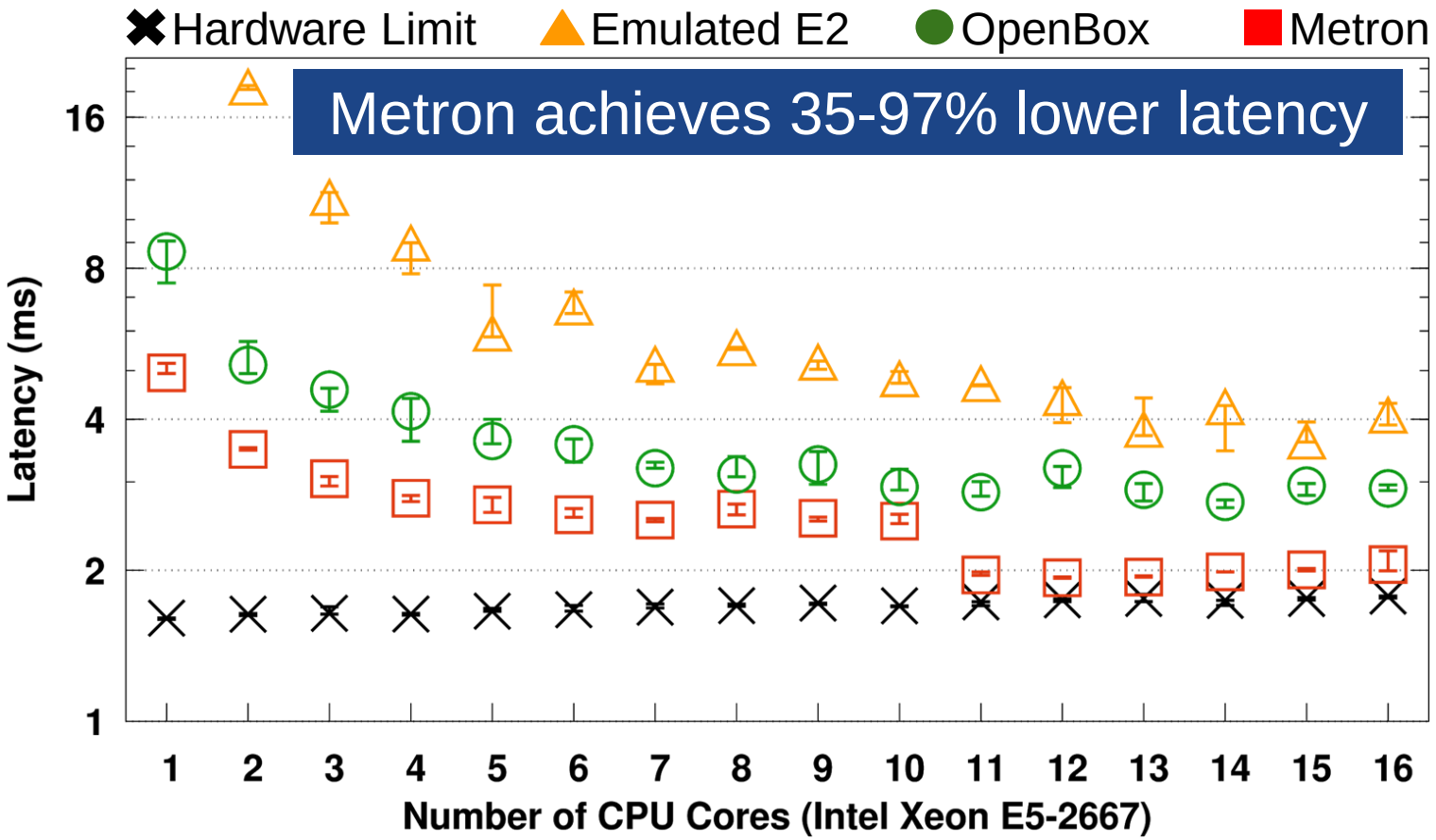
# Throughput at 40 Gbps



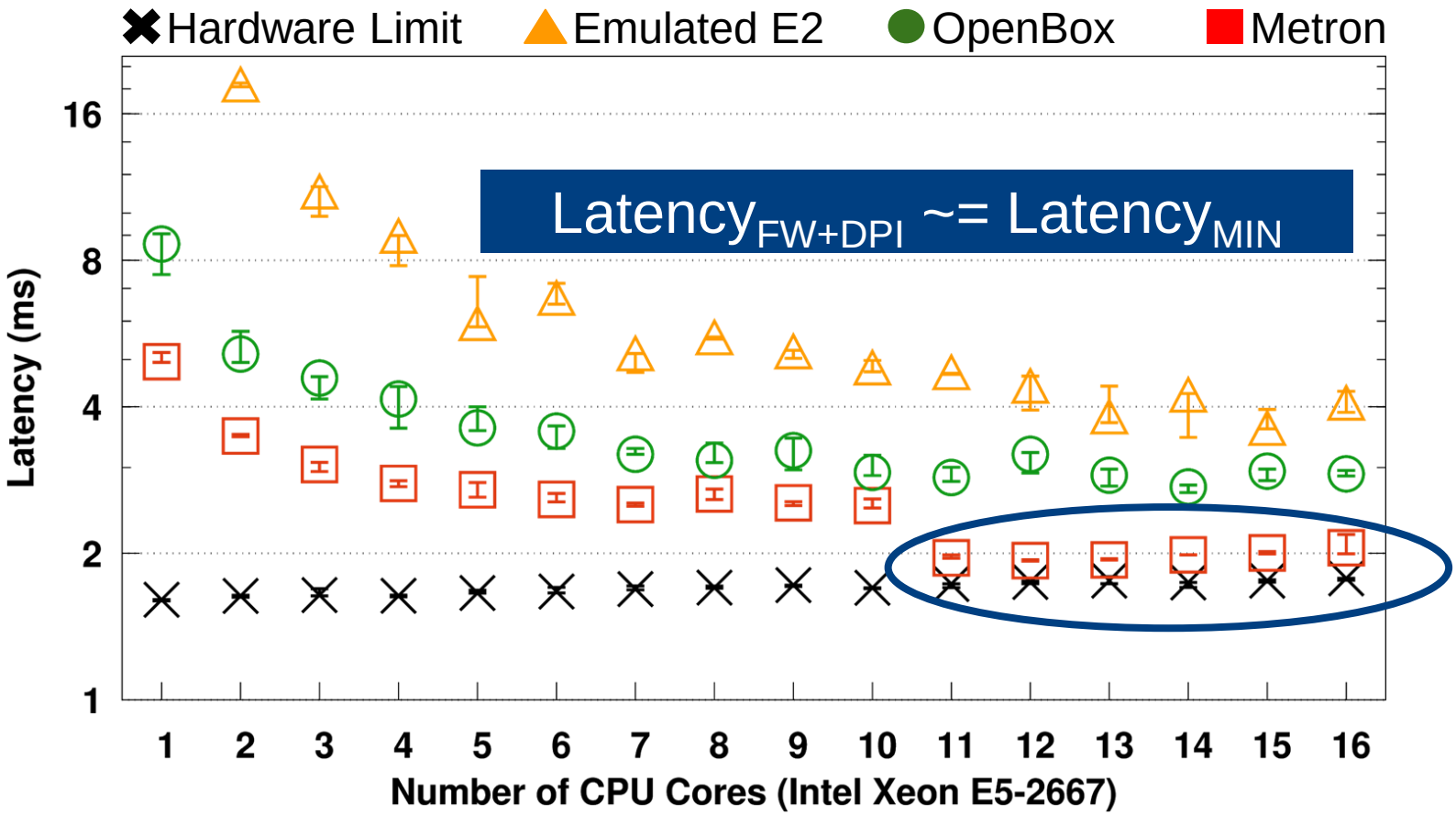
# Throughput at 40 Gbps



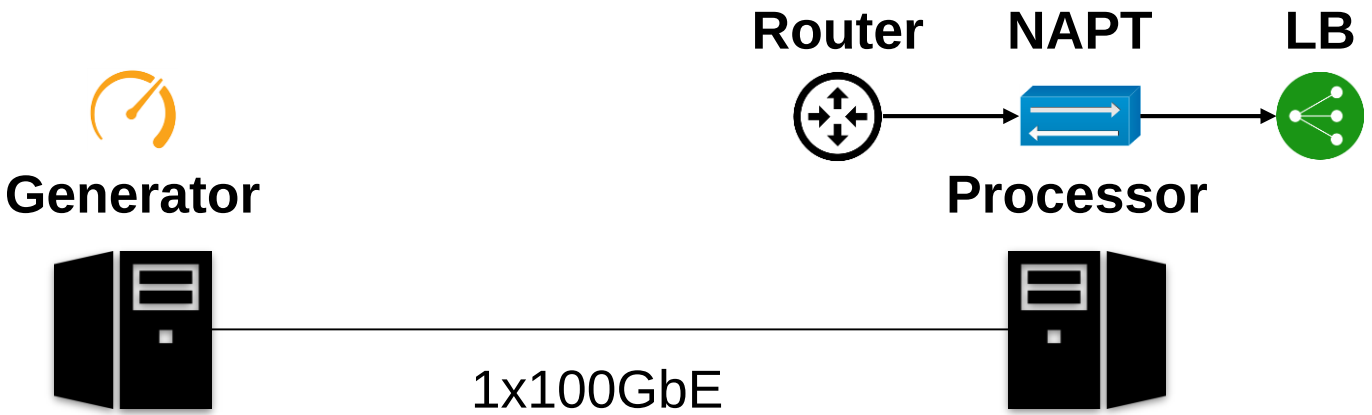
# Latency at 40 Gbps



# Latency at 40 Gbps

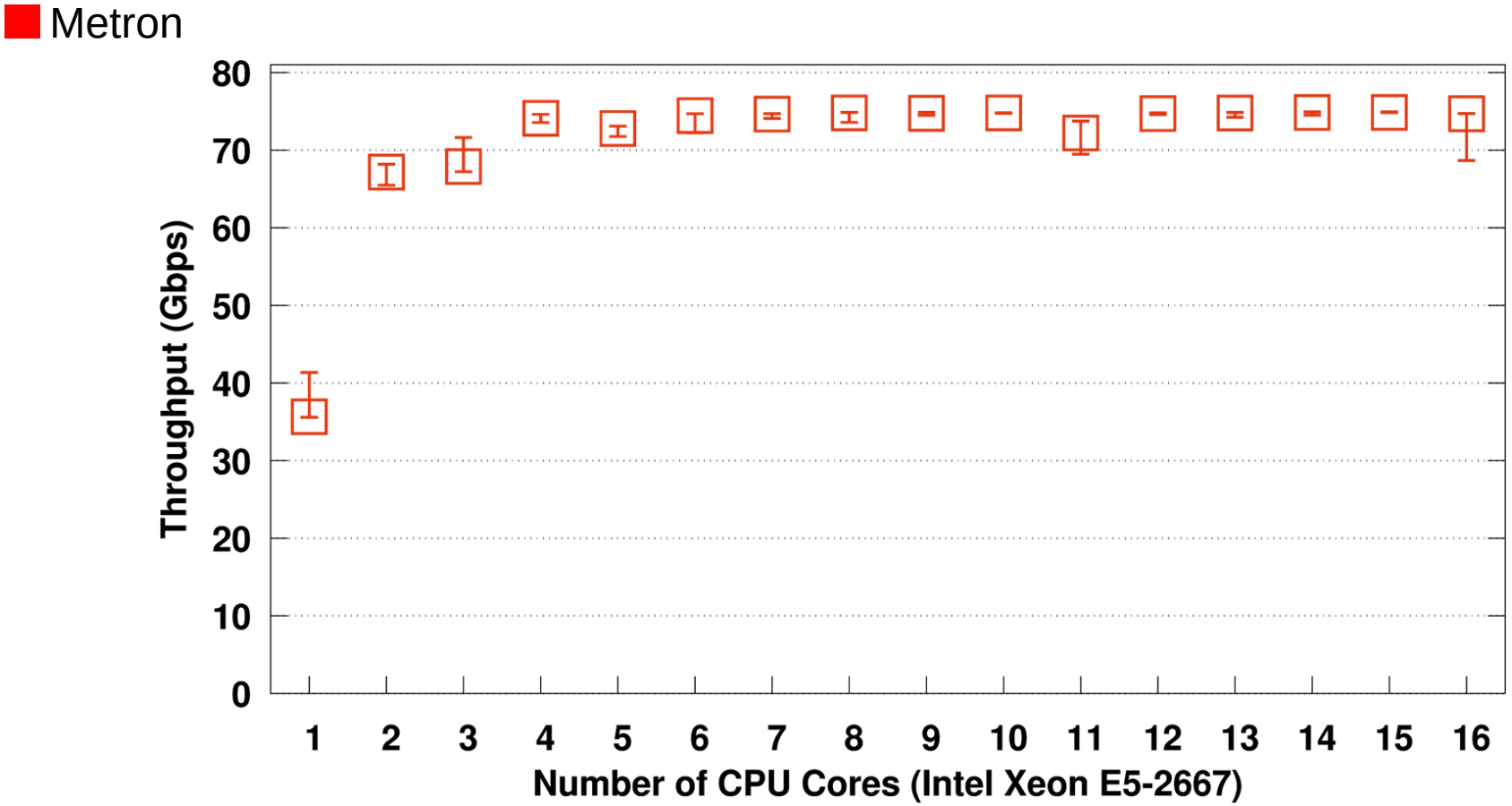


# Deployment at 100 Gbps

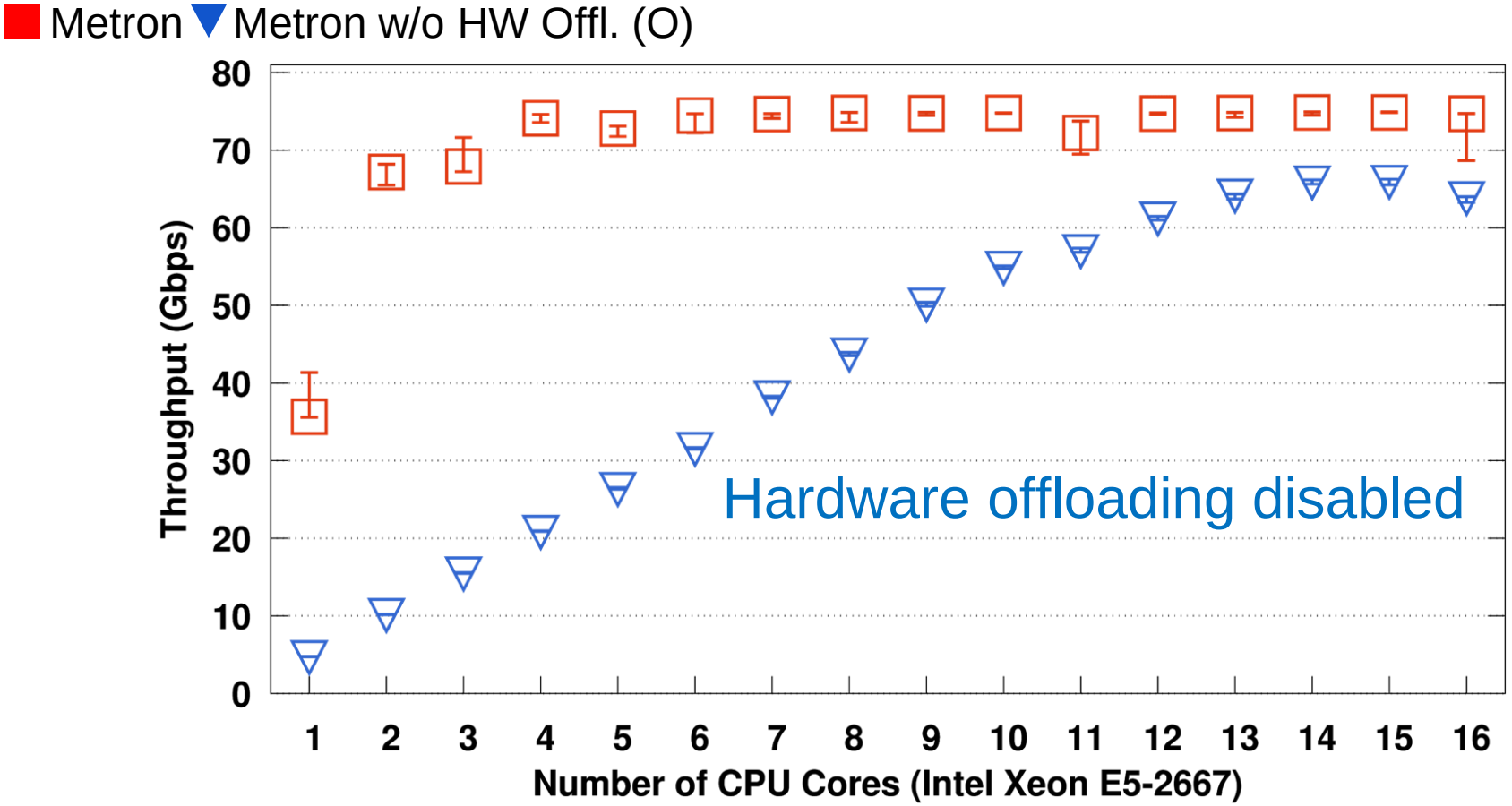



Total Throughput
100 Gbps

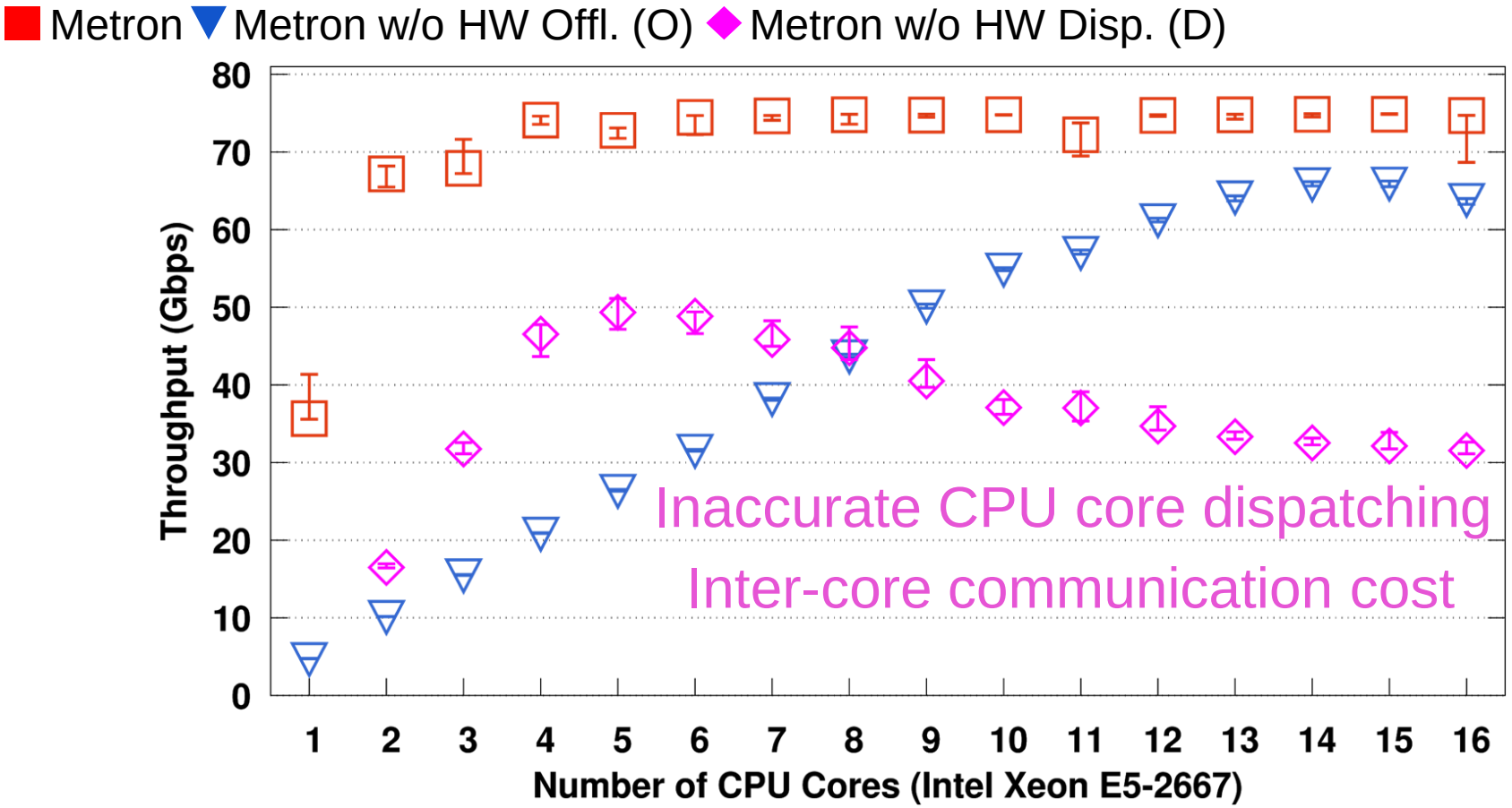
# Metron Decomposition at 100 Gbps



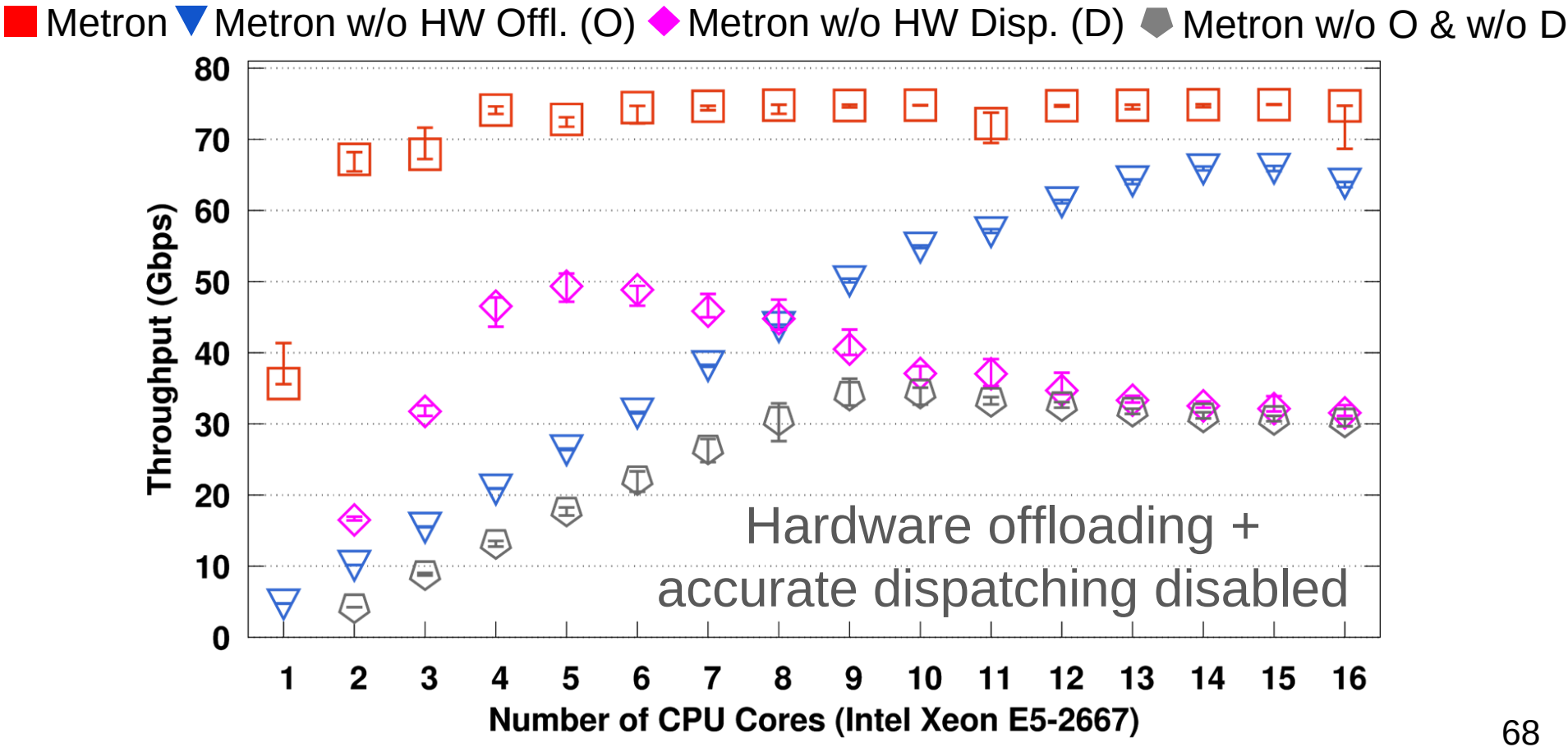
# Metron Decomposition at 100 Gbps



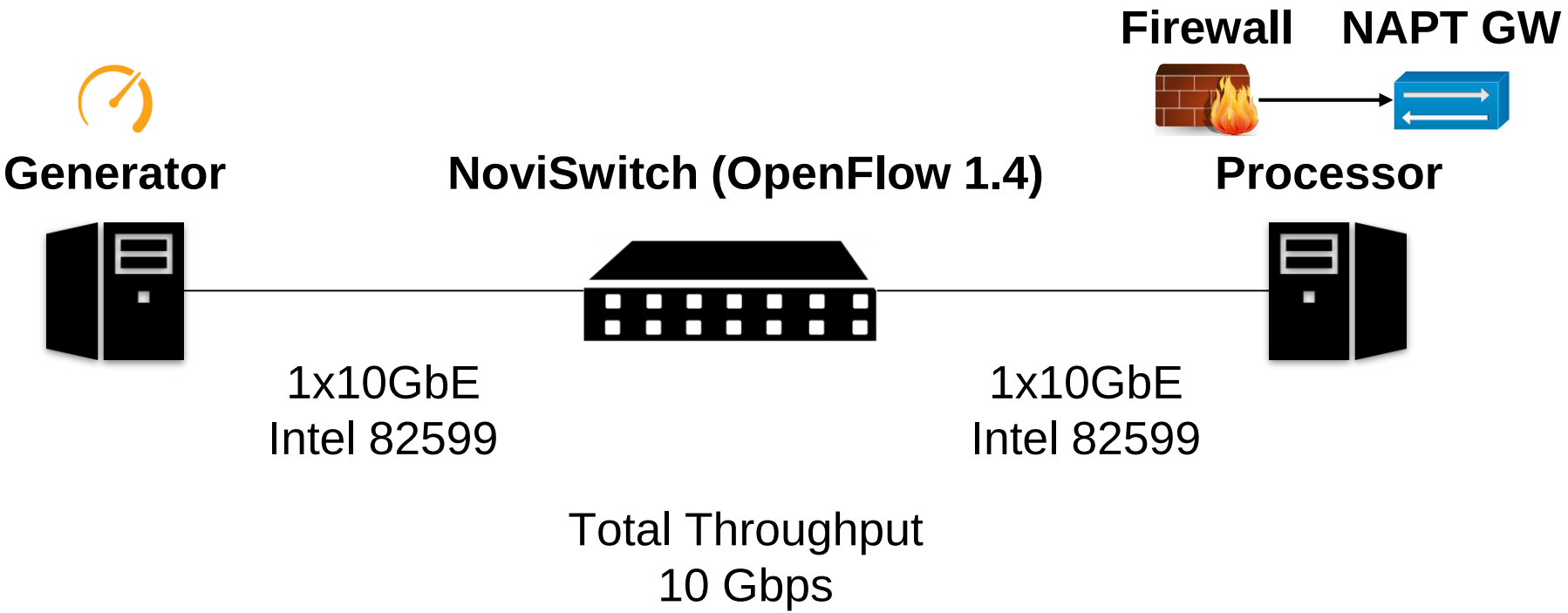
# Metron Decomposition at 100 Gbps



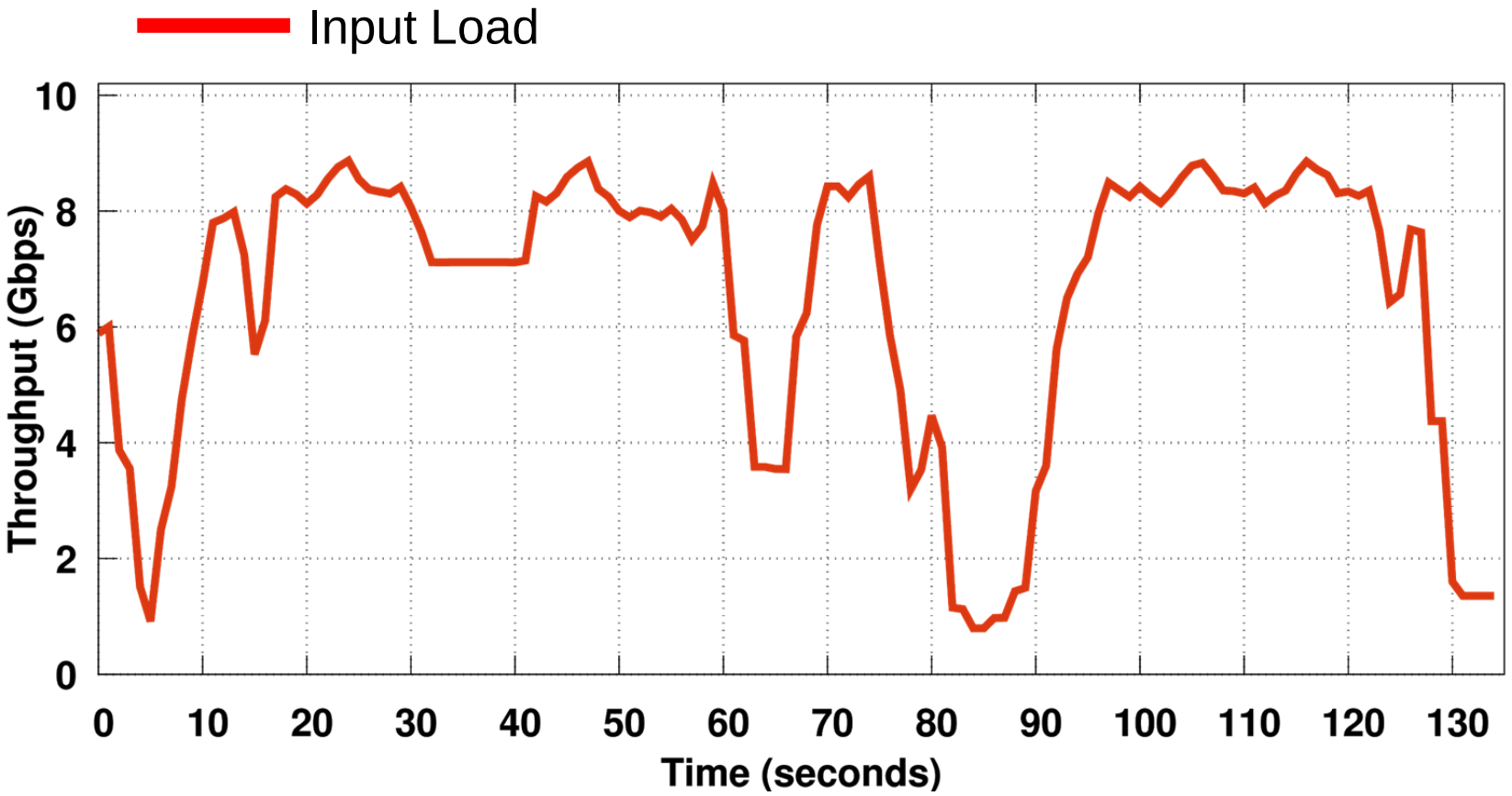
# Metron Decomposition at 100 Gbps



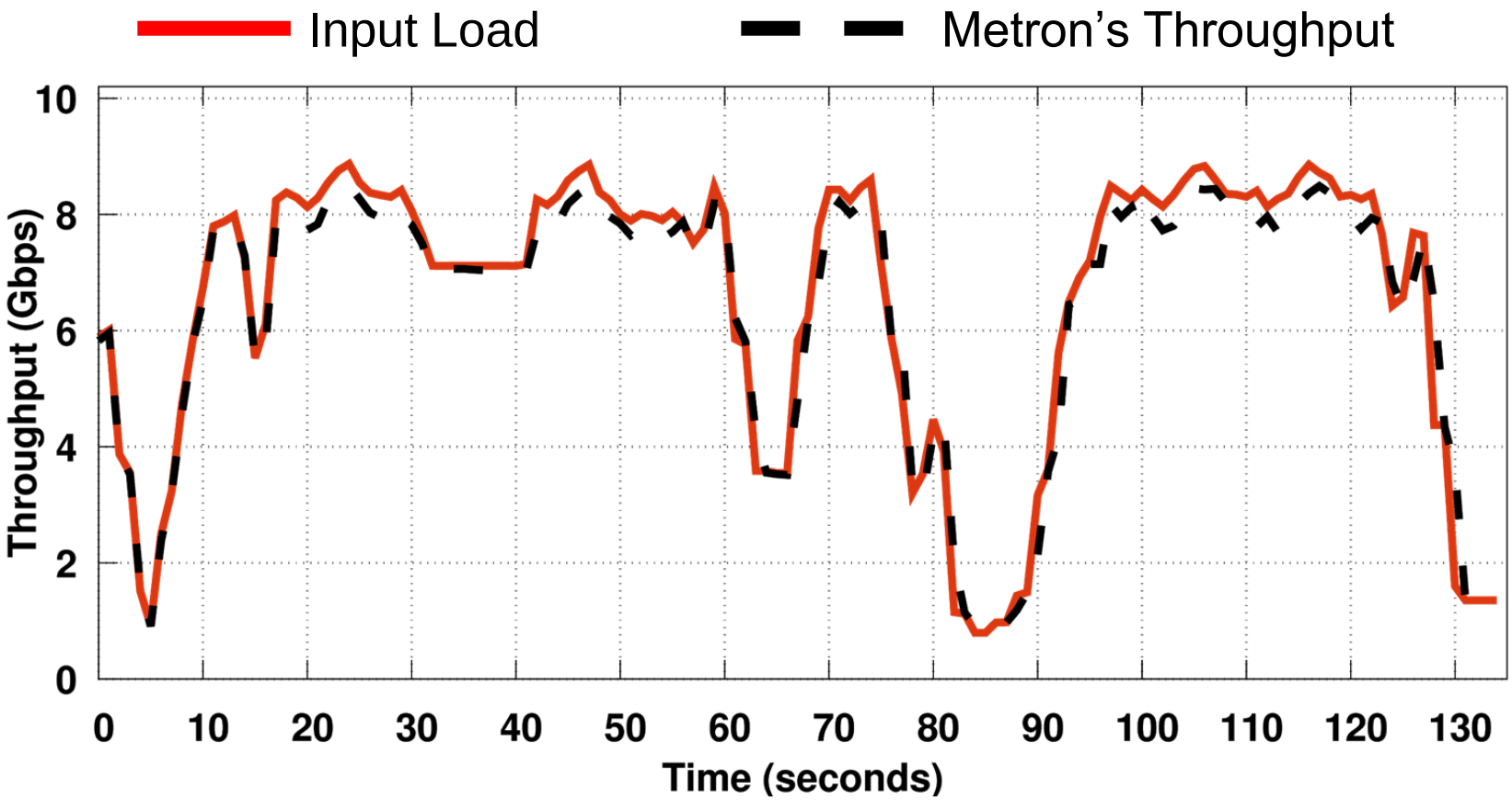
# Deployment at 10 Gbps



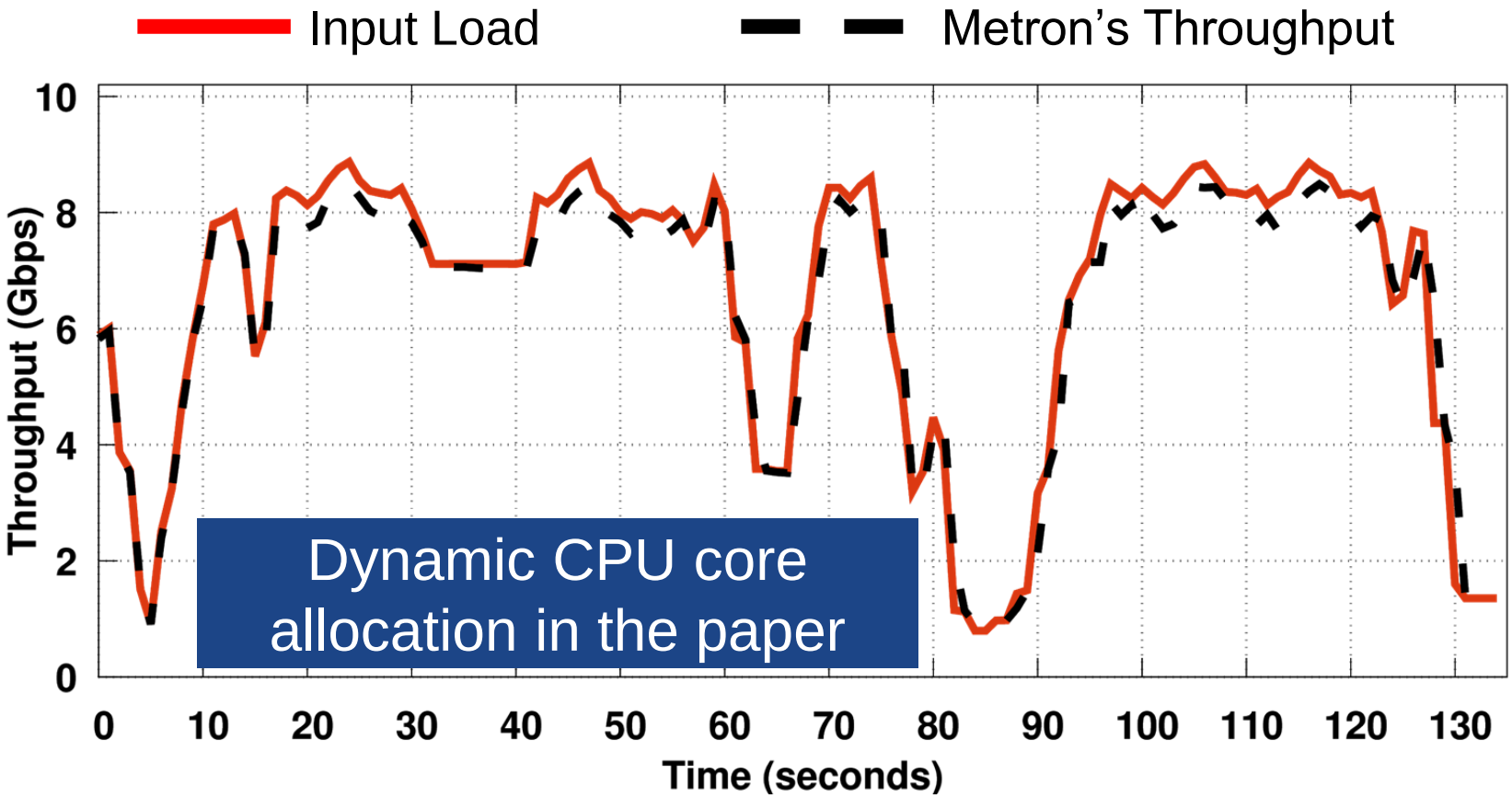
# Dynamic Load Balancing



# Dynamic Load Balancing



# Dynamic Load Balancing



# Open Source Activities

**Metron's driver for managing commodity servers is now in ONOS**

https://github.com/opennetworkinglab/onos/tree/master/drivers/server

https://github.com/gkatsikas/onos/tree/metron-driver

**Metron's high performance data plane extends FastClick**

https://github.com/tbarbette/fastclick/tree/metron

# Conclusion

## Metron

NFV service chains at the speed of the hardware

## Contributions

Orchestration of heterogeneous programmable hardware

Accurate CPU core dispatching

Hardware offloading

Quick and stable adaptation

Low-cost placement