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

Georgios P. Katsikas\textsuperscript{1,3}, Tom Barbette\textsuperscript{2}, Dejan Kostic\textsuperscript{3}, Rebecca Steinert\textsuperscript{1}, and Gerald Q. Maguire Jr.\textsuperscript{3}

\textsuperscript{1}RISE SICS \hspace{1cm} \textsuperscript{2}University of Liege \hspace{1cm} \textsuperscript{3}KTH Royal Institute of Technology
Introduction

High throughput and low latency
Service Chains of Network Functions

Router → NAT → IDS → Monitor → DPI → Router

Firewall

IPSec → LB

Web → Analytics → Video
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

The diagram shows the evolution of link speeds over time:

- 2009: 1 Gbps
- 2012: 10 Gbps
- 2015: 40 Gbps
- 2017: 50 Gbps
- Soon: 100 Gbps

The trend indicates a significant increase in link speeds from 2009 to 2017, with an expectation of reaching 100 Gbps in the near future.
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!

<table>
<thead>
<tr>
<th>Year</th>
<th>Link Speeds</th>
<th>Relative Increase in Transistor Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2012</td>
<td>10</td>
<td>10.5</td>
</tr>
<tr>
<td>2015</td>
<td>40</td>
<td>5</td>
</tr>
<tr>
<td>2017</td>
<td>50</td>
<td>7</td>
</tr>
<tr>
<td>Soon</td>
<td>100</td>
<td>10.5</td>
</tr>
</tbody>
</table>
Can existing NFV systems operate at these emerging link speeds (i.e., at 100 Gbps)?
Deployment at 100 Gbps

Generator → Processor

1x100GbE Mellanox Connect-X4

Router → NAPT → LB

Total Throughput
100 Gbps
Hardware Limit at 100 Gbps

Number of CPU Cores (Intel Xeon E5-2667)

Throughput (Gbps)
Existing NFV Solutions at 100 Gbps

- **Hardware Limit**
- **Emulated E2**

**Performance degradation**

- Throughput (Gbps) vs. Number of CPU Cores (Intel Xeon E5-2667)

- Data points indicate a trend of performance degradation as the number of CPU cores increases.
Existing NFV Solutions at 100 Gbps

- Hardware Limit
- Emulated E2
- OpenBox

Throughput (Gbps) vs. Number of CPU Cores (Intel Xeon E5-2667)
Metron – Hardware-level Performance

- Hardware Limit
- Emulated E2
- OpenBox
- Metron

Hardware offloading

Accurate CPU core dispatching

0 inter-core transfers
Metron – Hardware-level Performance

![Graph showing hardware performance comparison]

- Hardware Limit
- Emulated E2
- OpenBox
- Metron

76

Throughput (Gbps)

Number of CPU Cores (Intel Xeon E5-2667)

- 6.5x better efficiency
- Hardware performance
Why do existing NFV systems lose performance?
Software switch dispatching

Core 1  Core 2  Core 3  Core 4
Switch  NF1    NF2    Idle

E2, ClickOS, NetVM
Software switch dispatching
Software switch dispatching

- Core 1: Switch 1
- Core 2: NF1
- Core 3: NF2
- Core 4: Idle

E2, ClickOS, NetVM
Software switch dispatching

<table>
<thead>
<tr>
<th>Core 1</th>
<th>Core 2</th>
<th>Core 3</th>
<th>Core 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch</td>
<td>NF1</td>
<td>NF2</td>
<td>Idle</td>
</tr>
</tbody>
</table>

E2, ClickOS, NetVM
Software switch dispatching

Core 1: Switch
Core 2: NF1
Core 3: NF2
Core 4: Idle

Out
In
E2, ClickOS, NetVM
Software switch dispatching
Software switch dispatching

4 Inter-Core Transfers

Core 1: Switch
Core 2: NF1
Core 3: NF2
Core 4: Idle

E2, ClickOS, NetVM
<table>
<thead>
<tr>
<th>Core 1</th>
<th>Core 2</th>
<th>Core 3</th>
<th>Core 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch</td>
<td>NF1</td>
<td>NF2</td>
<td>Idle</td>
</tr>
</tbody>
</table>

4 Inter-Core Transfers

Pipeline dispatching with or without RSS

Software switch dispatching
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

- **Core 1**: Switch
- **Core 2**: NF1
- **Core 3**: NF2
- **Core 4**: Idle

---

Out

In

**Core 1**

- **Rx**

**Core 2**

- **Rx**

**Core 3**

- **NF1+NF2**

**Core 4**

- **Tx**

---

**Core 1**

- **Rx+NF1**

**Core 2**: Idle

**Core 3**: NF2+Tx

**Core 4**: Idle

---

**Flow Director**

Or

**RSS**

Out

In

**Core 1**

- **Out**

**Core 2**: E2, ClickOS, NetVM

**Core 3**: NFP, Flurries

**Core 4**: OpenBox, SNF, FastClick

---

**4 Inter-Core Transfers**

**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

Core 1
Switch

Core 2
NF1

Core 3
NF2

Core 4
Idle

Out
In

E2, ClickOS, NetVM

Rx

Rx

NF1+NF2

Tx

RSS

NFP, Flurries

Rx+NF1

Idle

NF2+Tx

Idle

Flow Director

Or

RSS

OpenBox, SNF, FastClick

4 Inter-Core Transfers

Pipeline dispatching with or without RSS

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

Core 1  Core 2  Core 3  Core 4
Switch  NF1  NF2  Idle

Out  In

E2, ClickOS, NetVM

Rx  Rx  NF1+NF2  Tx

Out  In

NFP, Flurries

Rx+NF1  Idle  NF2+Tx  Idle

Flow Director  Or

RSS  In

OpenBox, SNF, FastClick

4 Inter-Core Transfers

Rule or hash-based hardware dispatching

Core 1  Core 2  Core 3  Core 4
Switch  NF1  NF2  Idle

Out  In

E2, ClickOS, NetVM

Rx  Rx  NF1+NF2  Tx

Out  In

NFP, Flurries

Rx+NF1  Idle  NF2+Tx  Idle

Flow Director  Or

RSS  In

OpenBox, SNF, FastClick

4 Inter-Core Transfers

Rule or hash-based hardware dispatching

Core 1  Core 2  Core 3  Core 4
Switch  NF1  NF2  Idle

Out  In

E2, ClickOS, NetVM

Rx  Rx  NF1+NF2  Tx

Out  In

NFP, Flurries

Rx+NF1  Idle  NF2+Tx  Idle

Flow Director  Or

RSS  In

OpenBox, SNF, FastClick
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

Fastest system $\rightarrow$ fewest inter-core transfers

Rule or hash-based hardware dispatching
The available time to process 64-byte packets at 100 Gbps is \(~5\text{ns/packet}\)
The available time to process 64-byte packets at 100 Gbps is \(\sim 5\text{ns/packet}\).

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

Rule or hash-based hardware dispatching, with or without RSS.

1 Inter-Core Transfer.

OpenBox, SNF, FastClick.
The available time to process 64-byte packets at 100 Gbps is \(~5\text{ns/packet}\)

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

<table>
<thead>
<tr>
<th><strong>Problems</strong></th>
<th><strong>Challenges</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Greatly underutilized hardware</td>
<td>Hardware heterogeneity</td>
</tr>
<tr>
<td>Too many inter-core transfers</td>
<td>Dynamic adaptation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Solutions</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Unified hardware management abstractions</td>
</tr>
<tr>
<td>Dispatch traffic classes to CPU cores using tags. Map each tag to the correct core</td>
</tr>
</tbody>
</table>
# Zero Inter-core Transfers with Metron

<table>
<thead>
<tr>
<th>Problems</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greatly underutilized hardware</td>
<td>Hardware heterogeneity</td>
</tr>
<tr>
<td>Too many inter-core transfers</td>
<td>Dynamic adaptation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solutions</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unified hardware management abstractions</td>
<td>Up to 4.7x lower latency</td>
</tr>
<tr>
<td>Dispatch traffic classes to CPU cores using tags. Map each tag to the correct core</td>
<td>Up to 7.8x higher throughput</td>
</tr>
<tr>
<td></td>
<td>2.75-6.5x better efficiency</td>
</tr>
</tbody>
</table>
Firewall DPI

Input service chain to the controller
Katsikas et al.
SNF: synthesizing high performance NFV service chains
PeerJ Computer Science 2016

Input service chain to the controller

Synthesized NF

Source

Core 1

Core 2

Server
Input service chain to the controller

Source

Firewall DPI

Synthesized NF

Launcher

Read Ops

Write Ops

Splitter

Stateless

Stateful

Server

Core 1

Core 2

Metron Controller
Input service chain to the controller

Synthesized NF

Launcher
Read Ops
Write Ops

Splitter
Stateless
Stateful
Tagging

Source

Core 1
Core 2
 Server

Controller
Input service chain to the controller

Synthesized NF

Launcher

Read Ops

Write Ops

Splitter

Stateless

Stateful

Tagging

HW Rules

SW Ops

Source

Server

Core 1

Core 2
Input service chain to the controller

Synthesized NF

Launcher
Read Ops
Write Ops

Splitter
Stateless
Stateful
Tagging

HW Rules
SW Ops

Source

Server

Metron Controller

Core 1
Core 2

Firewall DPI

Read Ops
Write Ops

Stateless
Stateful

HW Rules
SW Ops
Firewall DPI

Input service chain to the controller

Collect run-time load statistics from switches and servers

Synthesized NF

Monitoring and Load Balancing

Launcher

Read Ops

Write Ops

Splitter

Stateless

Stateful

Tagging

HW Rules

SW Ops

Source

Tag 1

Core 1

Core 2

Server

Metron Controller
Firewall DPI
Input service chain to the controller
Collect run-time load statistics from switches and servers

Synthesized NF
Monitoring and Load Balancing
Detect Imbalance

Launcher
Read Ops
Write Ops

Splitter
Stateless
Stateful
Tagging
HW Rules
SW Ops

Core 1
Core 2
Overloaded Server

Source
Tag 1
Firewall DPI

Input service chain to the controller

Collect run-time load statistics from switches and servers

Synthesized NF

Monitoring and Load Balancing

Detect Imbalance

Split Traffic Classes

Launcher

Read Ops

Write Ops

Splitter

Stateless

Stateful

Tagging

HW Rules

SW Ops

Metron Controller

Source

Tag 1

Core 1

Core 2

Overloaded Server

HW Rules

SW Ops

Detect Imbalance

Split Traffic Classes

Tagging
Collect run-time load statistics from switches and servers.

Firewall DPI

Input service chain to the controller.

Synthesized NF

Monitoring and Load Balancing
- Detect Imbalance
- Split
- Traffic Classes
- Update

Launcher
- Read Ops
- Write Ops

Splitter
- Stateless
- Stateful

Tagging

HW Rules

SW Ops

Source

Core 1

Core 2

Overloaded Server
Input service chain to the controller
Collect run-time load statistics from switches and servers

Synthesized NF

Monitoring and Load Balancing
- Detect Imbalance
- Split Traffic Classes
- Update

Launcher
Read Ops
- Splitter
- Stateless
- Tagging
- HW Rules

Write Ops
- Stateful
- SW Ops

Metron Controller

Core 1

Core 2

Overloaded Server

Tags 1

Source
Firewall DPI
Input service chain to the controller
Collect run-time load statistics from switches and servers

Synthesized NF
Monitoring and Load Balancing
Detect Imbalance
Split
Traffic Classes
Update

Launcher
Read Ops
Write Ops
Splitter
Stateless
Tagging
Stateful
HW Rules
SW Ops

Source
Tags 1, 2
Core 1
Core 2
Balanced Server

Metron Controller

Read Ops
Write Ops

Splitter
Stateless
Tagging
Stateful
HW Rules
SW Ops

52
Performance Evaluation
Deployment at 40 Gbps

Generator

NoviSwitch (OpenFlow 1.4)

Total Throughput
40 Gbps

Processor

Firewall

DPI

4x10GbE
Intel 82599

4x10GbE
Intel 82599
Throughput at 40 Gbps

Hardware Limit

38

Number of CPU Cores (Intel Xeon E5-2667)
Throughput at 40 Gbps

- Hardware Limit
- Emulated E2

\[ \sim 1.70 \text{ Gbps/core, 16 cores, } R^2=0.94 \]
Throughput at 40 Gbps

- Hardware Limit
- Emulated E2
- OpenBox

~2.08 Gbps/core, 11 cores, $R^2=0.92$
Throughput at 40 Gbps

- Hardware Limit
- Emulated E2
- OpenBox
- Metron

~4.0 Gbps/core, 10 cores, $R^2=0.86$
Throughput at 40 Gbps

- **Hardware Limit**
- **Emulated E2**
- **OpenBox**
- **Metron**

~3x better efficiency
Throughput at 40 Gbps

- Hardware Limit
- Emulated E2
- OpenBox
- Metron

DPI cost

Throughput (Gbps)

Number of CPU Cores (Intel Xeon E5-2667)
FW + DPI at the speed of a 40 Gbps testbed
Latency at 40 Gbps

Metron achieves 35-97% lower latency

- Hardware Limit
- Emulated E2
- OpenBox
- Metron

Number of CPU Cores (Intel Xeon E5-2667)
Latency at 40 Gbps

Latency_{FW+DPI} \approx \text{Latency}_{\text{MIN}}
Deployment at 100 Gbps

1x100GbE Mellanox Connect-X4

Total Throughput
100 Gbps
Metron Decomposition at 100 Gbps

Throughput (Gbps)

Number of CPU Cores (Intel Xeon E5-2667)
Metron Decomposition at 100 Gbps

Hardware offloading disabled
Metron Decomposition at 100 Gbps

Inaccurate CPU core dispatching
Inter-core communication cost
Metron Decomposition at 100 Gbps

- Metron
- Metron w/o HW Offl. (O)
- Metron w/o HW Disp. (D)
- Metron w/o O & w/o D

Hardware offloading + accurate dispatching disabled
Deployment at 10 Gbps

Generator

NoviSwitch (OpenFlow 1.4)

Processor

Firewall

NAPT GW

1x10GbE
Intel 82599

1x10GbE
Intel 82599

Total Throughput
10 Gbps
Dynamic Load Balancing

Input Load

Throughput (Gbps)

Time (seconds)
Dynamic Load Balancing

- Input Load
- Metron’s Throughput

![Graph showing dynamic load balancing](image)
Dynamic Load Balancing

- Input Load
- Metron’s Throughput

Dynamic CPU core allocation in the paper
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
- Quick and stable adaptation
- Hardware offloading
- Low-cost placement