Network Functions

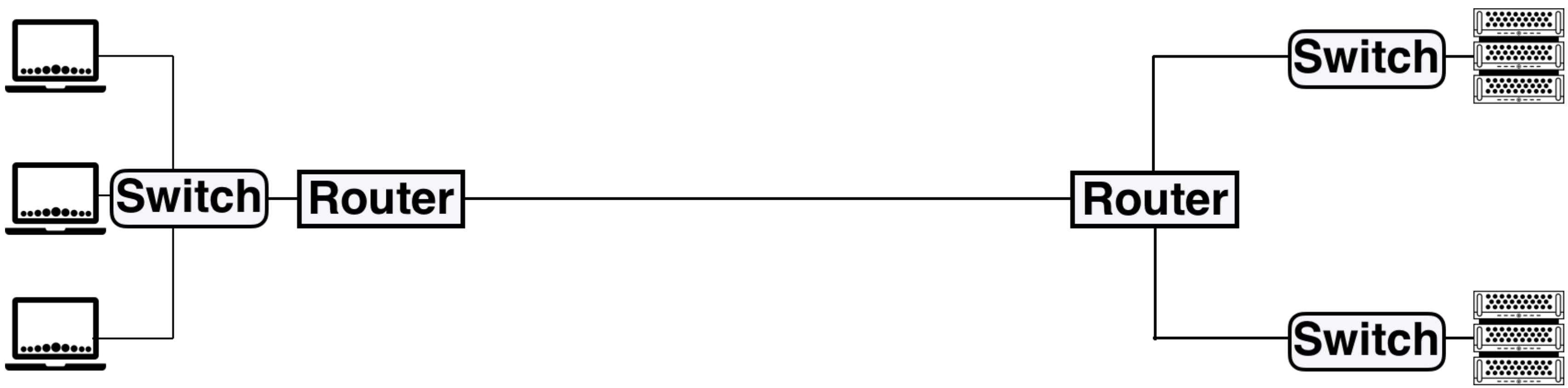
ECE/CS598HPN

Radhika Mittal

Logistics

- Tuesday, Dec 1st: Students' presentation (and choice)
  - Sign up for a paper by the end of this week.
- Thursday, Dec 3rd: No reading assignment, only wrap-up lecture.
- Friday, Dec 4th: Final project report due.
- Tuesday, Dec 8th: Project presentation. Details TBA.

Conventional view of networks

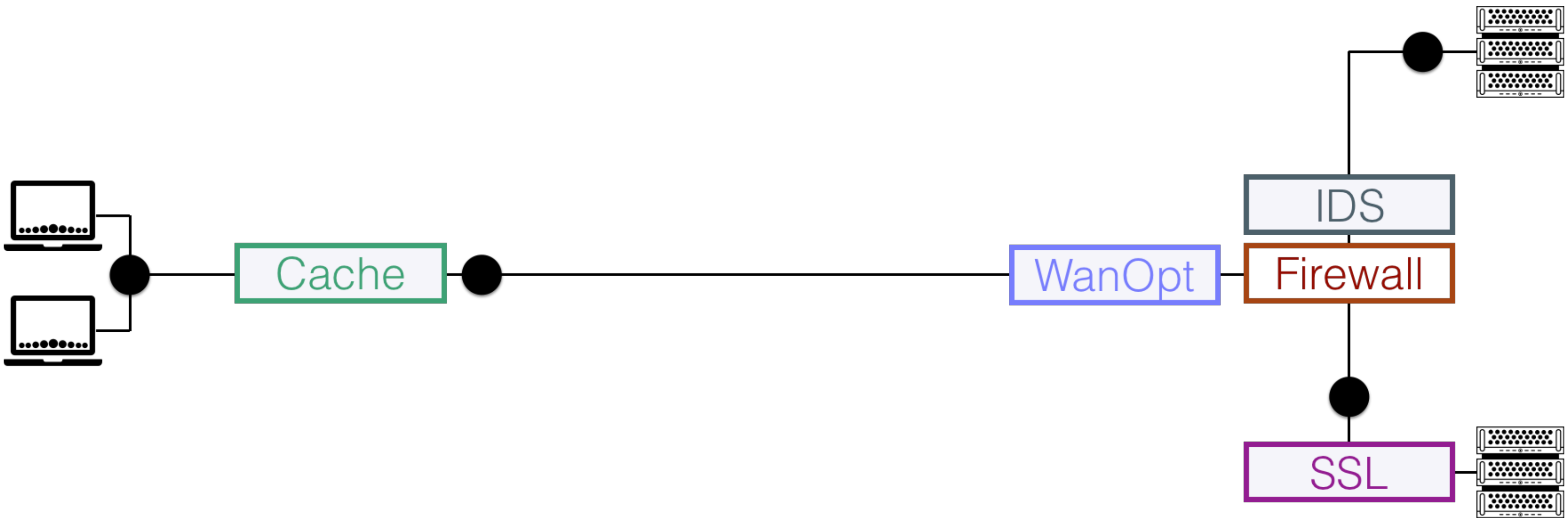


Data delivery is the only functionality provided by such a network.

**contents of the slide borrowed from talks given by Aurojit Panda, NYU*

Rise of middleboxes

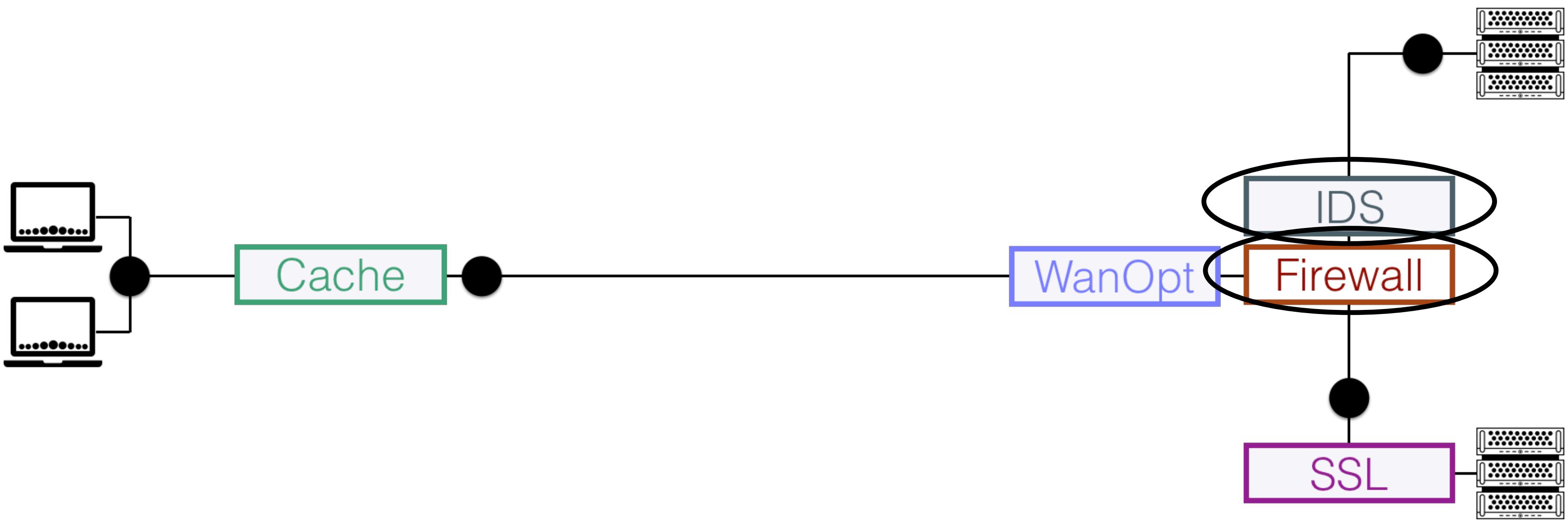
Data delivery is not the only required functionality.



**contents of the slide borrowed from talks given by Aurojit Panda, NYU*

Rise of middleboxes

Data delivery is not the only required functionality.

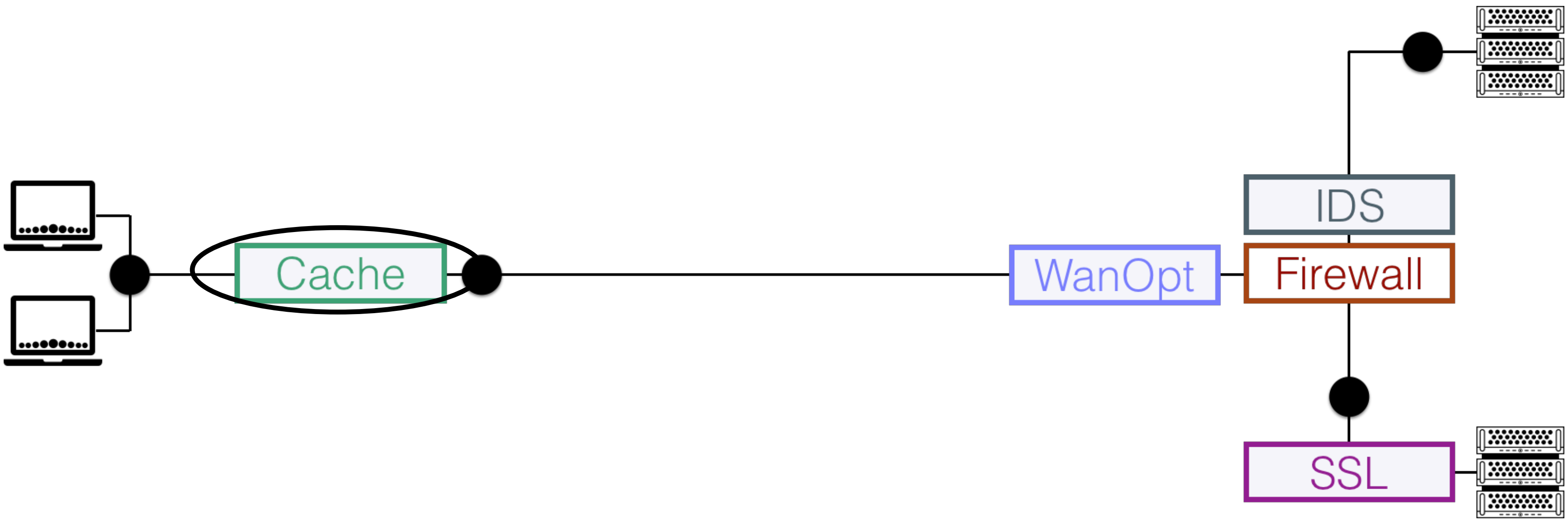


Security: identify and block unwanted traffic.

**contents of the slide borrowed from talks given by Aurojit Panda, NYU*

Rise of middleboxes

Data delivery is not the only required functionality.

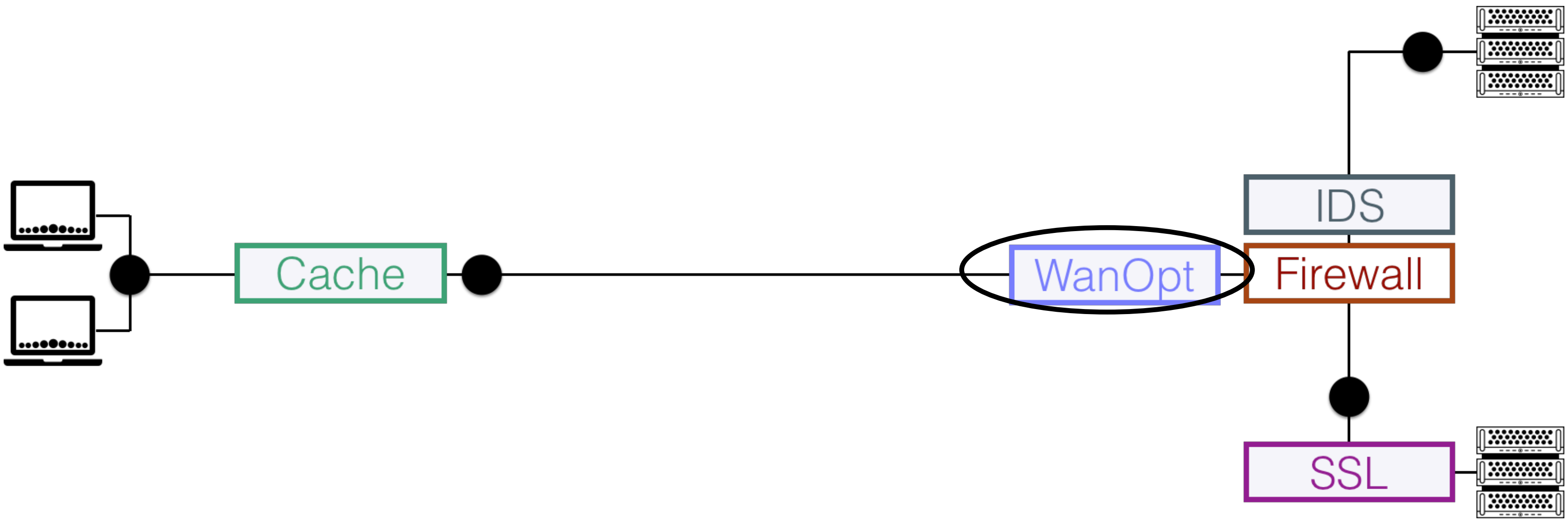


Performance: load content faster.

**contents of the slide borrowed from talks given by Aurojit Panda, NYU*

Rise of middleboxes

Data delivery is not the only required functionality.

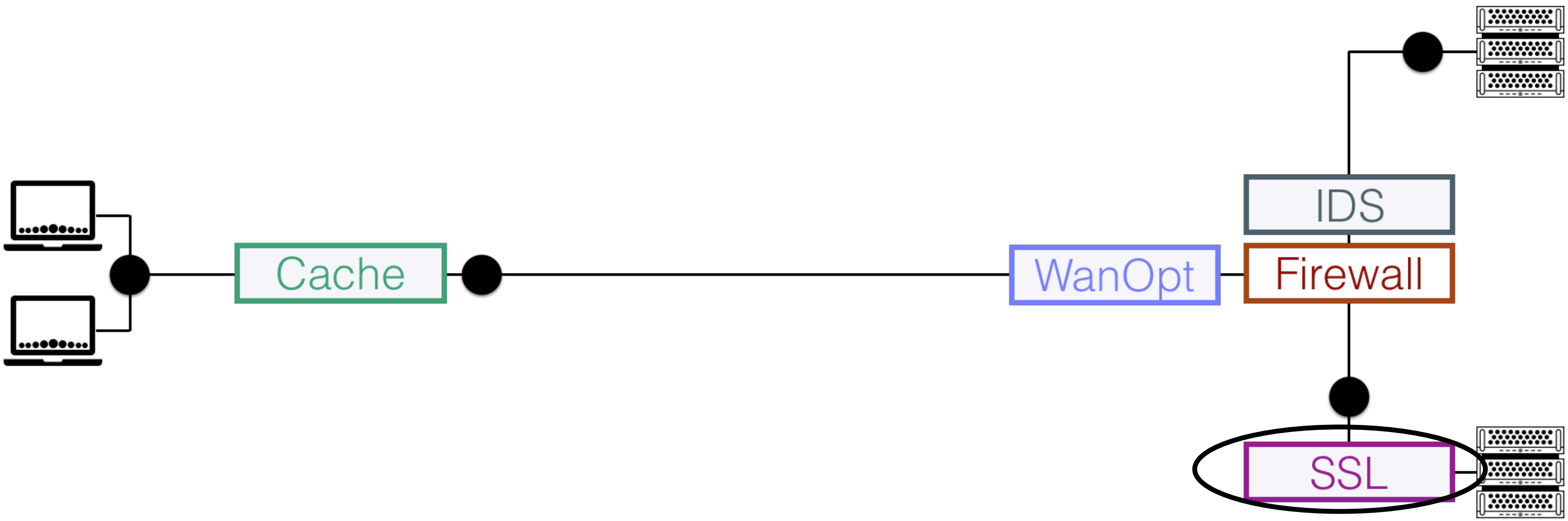


Performance: reduce bandwidth usage.

**contents of the slide borrowed from talks given by Aurojit Panda, NYU*

Rise of middleboxes

Data delivery is not the only required functionality.

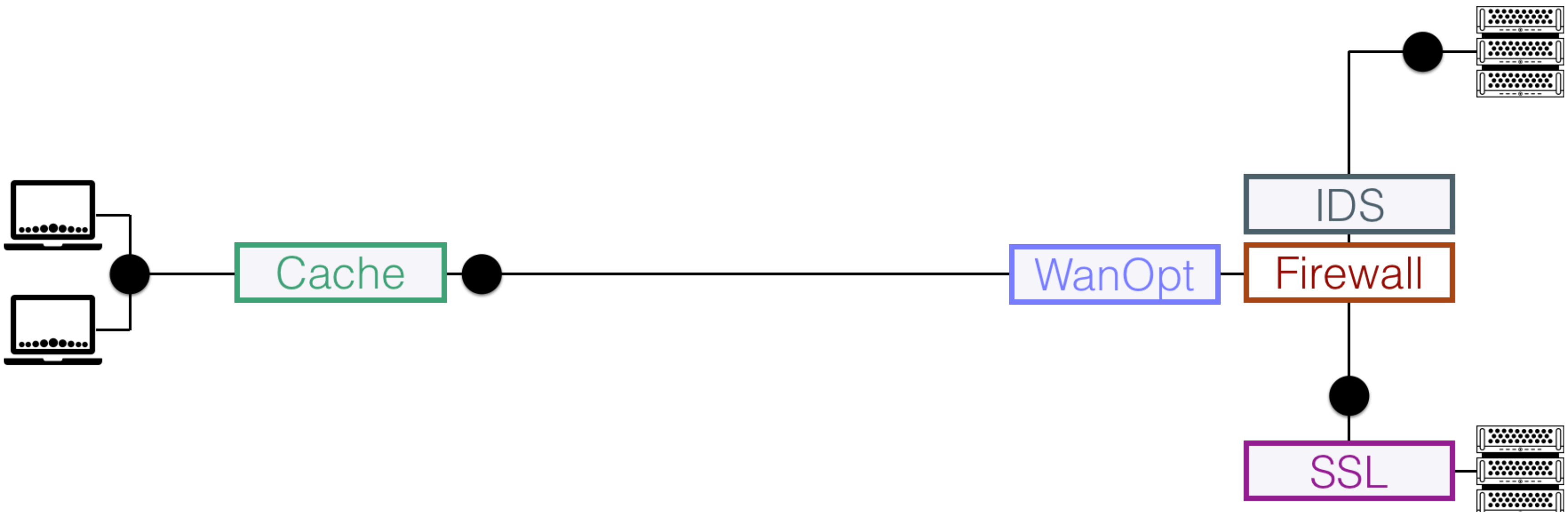


Application support: protocol for legacy application.

**contents of the slide borrowed from talks given by Aurojit Panda, NYU*

Rise of middleboxes

Data delivery is not the only required functionality.



One-third of all network devices in enterprises are middleboxes!

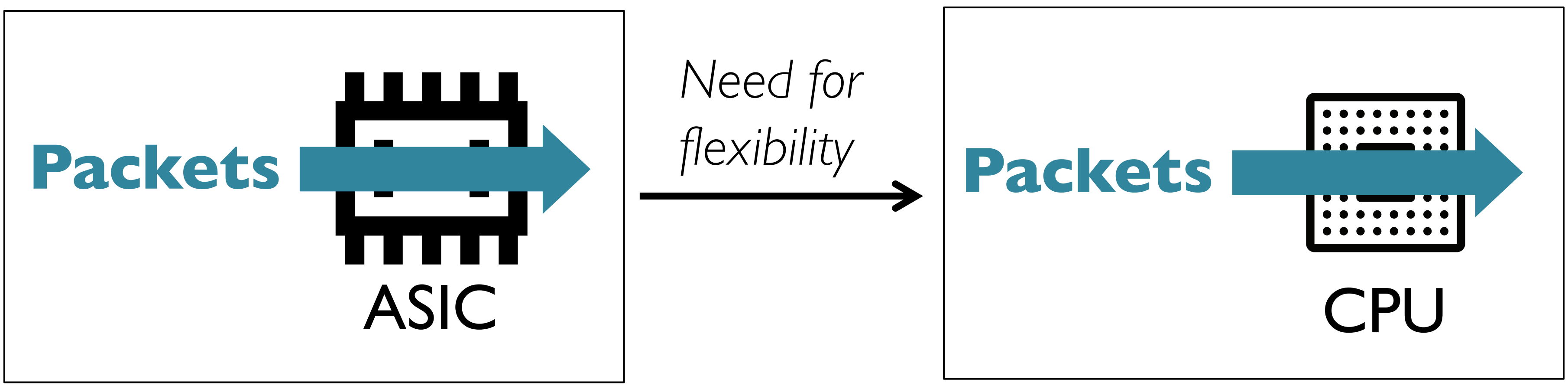
(source: Sherry et. al., SIGCOMM'12)

**contents of the slide borrowed from talks given by Aurojit Panda, NYU*

Evolution of middleboxes

Dedicated hardware

Software

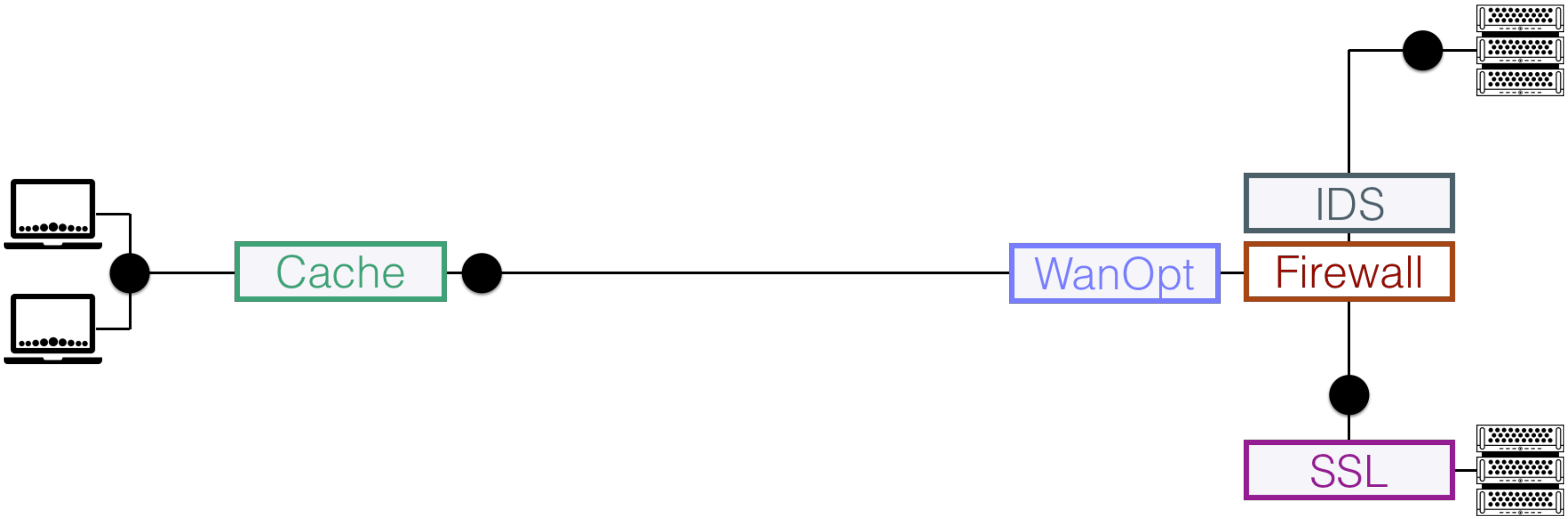


Middleboxes

Network functions

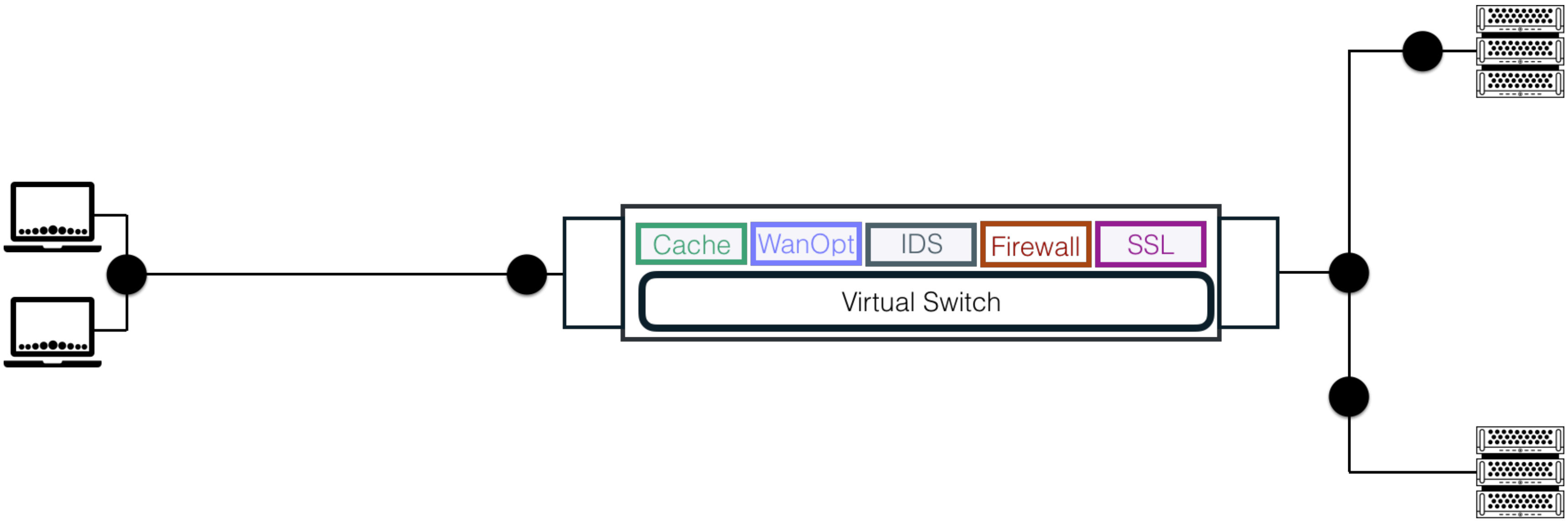
**contents of the slide borrowed from talks given by Aurojit Panda, NYU*

From hardware middleboxes….



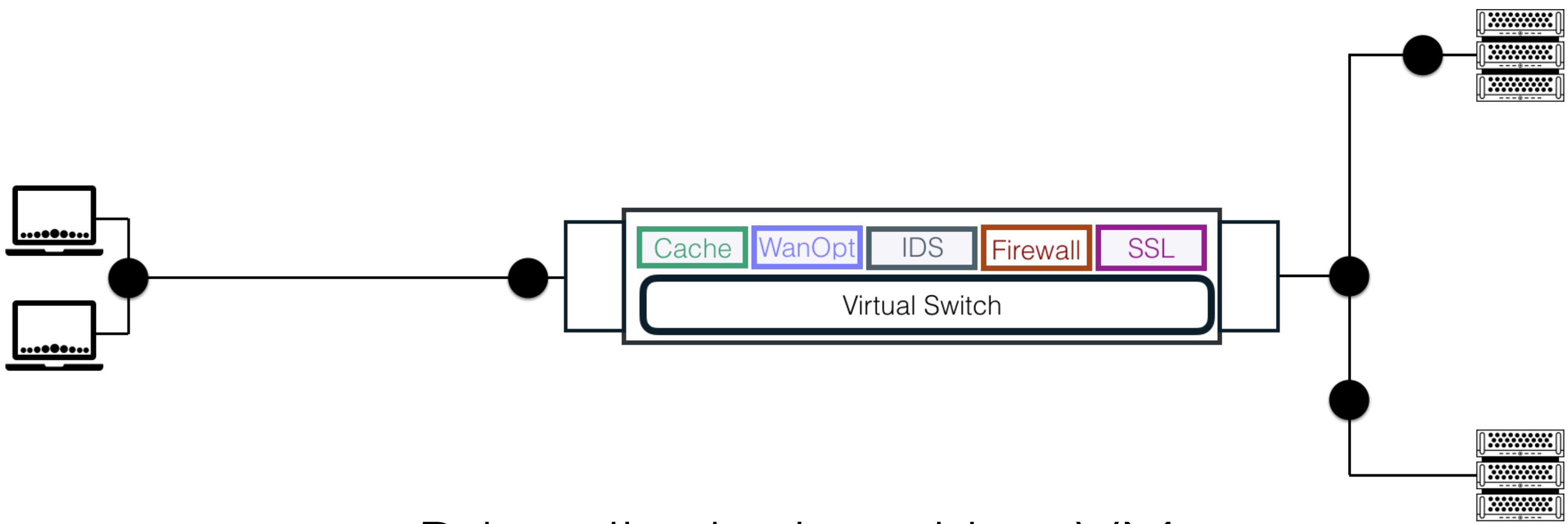
**contents of the slide borrowed from talks given by Aurojit Panda, NYU*

…to software network functions (NFs)



**contents of the slide borrowed from talks given by Aurojit Panda, NYU*

…to software network functions (NFs)



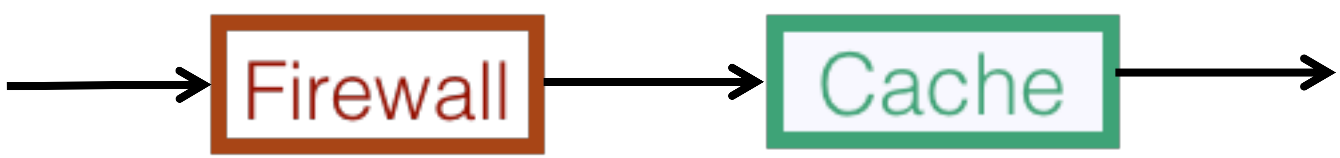
Primarily deployed in a VM

(Network Function Virtualization)

**contents of the slide borrowed from talks given by Aurojit Panda, NYU*

Key benefits of software network functions

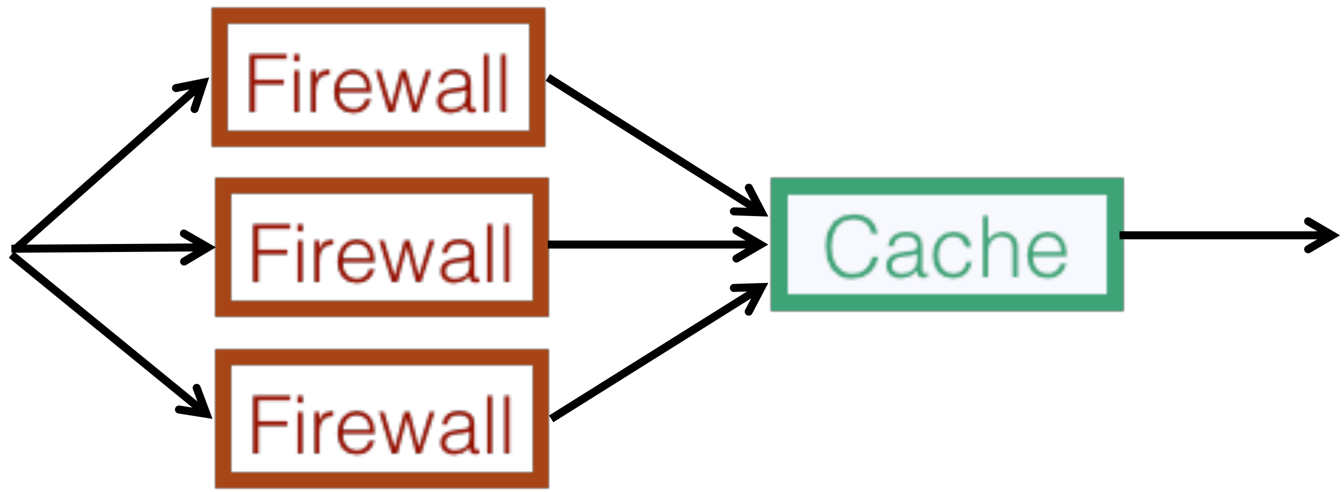
- Programmability
  - ability to update and create new NFs.
- Ease of deployment, configuration, and management.



NF Service Chain

Key benefits of software network functions

- Programmability
  - ability to update and create new NFs.
- Ease of deployment, configuration, and management.



Key benefits of software network functions

- Programmability
  - ability to update and create new NFs.
- Ease of deployment, configuration, and management.

Being adopted by both carriers and cloud providers.

Benefits of software NF come at a cost

- Complex and costly state management.
- Unpredictable performance.
- Performance degradation.

State management during scaling or failover

Split/Merge: System Support for Elastic Execution in Virtual Middleboxes

Shriram Rajagopalan[†‡], Dan Williams[†], Hani Jamjoom[†], and Andrew Warfield[‡]
[†]IBM T. J. Watson Research Center, Yorktown Heights, NY
[‡]University of British Columbia, Vancouver, Canada

Elastic Scaling of Stateful Network Functions

Shinae Woo[*†]*, Justine Sherry*[‡]*, Sangjin Han*[*]*, Sue Moon*[†]*, Sylvia Ratnasamy*[*]*, and Scott Shenker*[*§]
[*]*University of California, Berkeley* [†]*KAIST* [‡]*CMU* [§]*ICSI*

Stateless Network Functions: Breaking the Tight Coupling of State and Processing

Murad Kablan, Azzam Alsudais, Eric Keller
University of Colorado, Boulder
Franck Le
IBM Research

Rollback-Recovery for Middleboxes

Justine Sherry[*] Peter Xiang Gao[*] Soumya Basu[*] Aurojit Panda[*]
Arvind Krishnamurthy[•] Christian Maciocco[†] Maziar Manesh[†] João Martins[◁]
Sylvia Ratnasamy[*] Luigi Rizzo[‡] Scott Shenker[○*]
[*] UC Berkeley [•] University of Washington [†] Intel Research

Pico Replication: A High Availability Framework for Middleboxes

Shriram Rajagopalan[†‡] Dan Williams[†] Hani Jamjoom[†]
[†]IBM T. J. Watson Research Center, Yorktown Heights
[‡]University of British Columbia

OpenBox: A Software-Defined Framework for Developing, Deploying, and Managing Network Functions

Anat Bremler-Barr [*]
bremler@idc.ac.il
Yotam Harchol [†]
yotamhc@cs.huji.ac.il
David Hay[†]
dhay@cs.huji.ac.il
[*] School of Computer Science, The Interdisciplinary Center, Herzliya, Israel
[†] School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel

E2: A Framework for NFV Applications

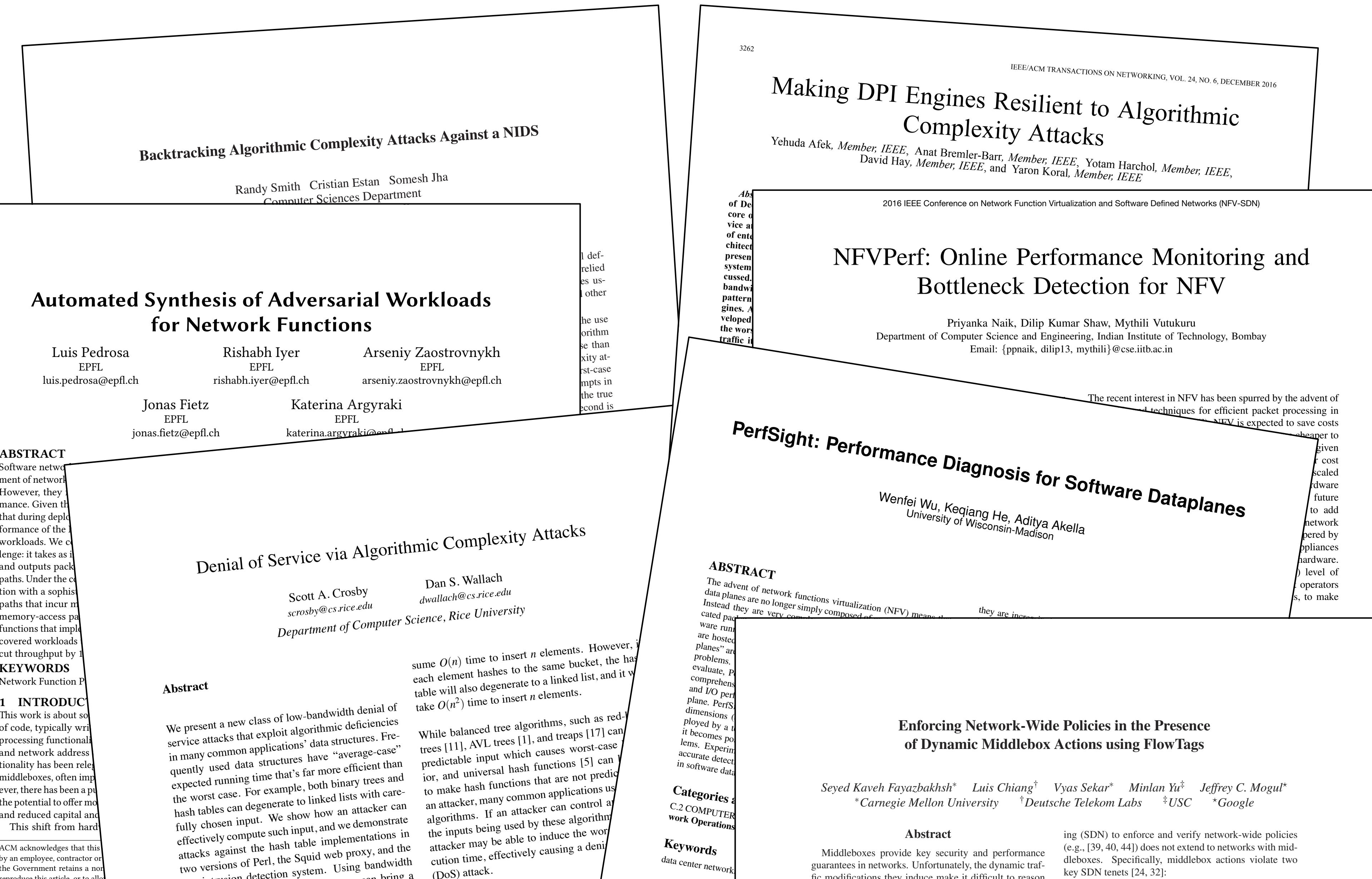
Shoumik Palkar[*], UC Berkeley, sppalkar@berkeley.edu
Chang Lan[*], UC Berkeley, clan@eecs.berkeley.edu
Sangjin Han, UC Berkeley, sangjin@eecs.berkeley.edu
Keon Jang, Intel Labs, keon.jang@intel.com
Aurojit Panda, UC Berkeley, apanda@cs.berkeley.edu
Sylvia Ratnasamy, UC Berkeley, sylvia@eecs.berkeley.edu
Luigi Rizzo, Università di Pisa, rizzo@iet.unipi.it
Scott Shenker, UC Berkeley and ICSI, shenker@icsi.berkeley.edu

Paving the Way for NFV: Simplifying Middlebox Modifications using StateAlyzr

Junaid Khalid, Aaron Gember-Jacobson, Roney Michael, Anubhavnidhi Abhashkumar, Aditya Akella
University of Wisconsin-Madison

Abstract

Understanding NF Performance



Providing guarantees about NF behavior

High performance NF implementations

Microboxes: High Performance NFV with Customizable, Asynchronous TCP Stacks and Dynamic Subscriptions

Guyue Liu*, Yuxin Ren*, Mykola Yurchenko*,
K.K. Ramakrishnan†, Timothy Wood*
*George Washington University, †University of California, Riverside

mOS: A Reusable Networking Stack for Flow Monitoring Middleboxes

Muhammad Jamshed, YoungGyoun Moon, Donghwi Kim, Dongsu Han, and KyoungSoo Park
School of Electrical Engineering, KAIST

FlowBlaze: Stateful Packet Processing in Hardware

Salvatore Pontarelli[1,2], Roberto Bifulco[3], Marco Bonola[1,2], Carmelo Cascone[4], Marco Spaziani[2,5], Valerio Bruschi[2,5], Davide Sanvito[6], Giuseppe Siracusano[3], Antonio Capone[6], Michio Honda[3], Felipe Huici[3] and Giuseppe Bianchi[2,5]

[1]Axbryd, [2]CNIT, [3]NEC Laboratories Europe, [4]Open Networking Foundation, [5]University of Rome Tor Vergata, [6]Politecnico di Milano

Abstract

While programmab
handle growing net
yet simple abstracti
in hardware remain
problem with Flow
stateful packet pro
straction is based o
troduces the explici
Blaze to leverage f
pressive, supporting
tions, and easy to u
tation issues from t
FlowBlaze on a Ne
tency (in the order
tively little power,
thousands of flows,
for even higher spe
ware and software
licly available.

1 Introductio

Network infrastruct
network functions t
and server load bala
such as access con
examples. Given
the need to contin
tors have turned to

ClickNP: Highly Flexible and High Performance Network Processing with Reconfigurable Hardware

Bojie Li§† Kun Tan† Layong (Larry) Luo‡ Yanqing Peng•† Renqian Luo§†
Ningyi Xu† Yongqiang Xiong† Peng Cheng† Enhong Chen§
†Microsoft Research §USTC ‡Microsoft •SJTU

RACT

lexible software network functions (NFs) are cru-
ponents to enable multi-tenancy in the clouds. How-
tware packet processing on a commodity server has
capacity and induces high latency. While software
ld scale out using more servers, doing so adds sig-
cost. This paper focuses on accelerating NFs with
mable hardware, *i.e.*, FPGA, which is now a ma-
nology and inexpensive for datacenters. However,
predominately programmed using low-level hard-
cription languages (HDLs), which are hard to code
cult to debug. More importantly, HDLs are almost
ible for most software programmers. This paper presents
a FPGA-accelerated platform for highly flexible
-performance NFs with commodity servers. ClickNP
flexible as it is completely programmable using
l C-like languages, and exposes a modular program-
traction that resembles Click Modular Router. ClickNP
igh performance. Our prototype NFs show that they
ess traffic at up to 200 million packets per second
a-low latency (< 2μs). Compared to existing soft-
nterparts, with FPGA, ClickNP improves through-
0x, while reducing latency by 10x. To the best of
vledge, ClickNP is the first FPGA-accelerated plat-
NFs, written completely in high-level language and
g 40 Gbps line rate at any packet size.

oncepts

1. INTRODUCTION

Modern multi-tenant datacenters provide shared infrastructure for hosting many different types of services from different customers (*i.e.*, tenants) at a low cost. To ensure security and performance isolation, each tenant is deployed in a *virtualized network* environment. Flexible network functions (NFs) are required for datacenter operators to enforce isolation while simultaneously guaranteeing Service Level Agreements (SLAs).

Conventional hardware-based network appliances are not flexible, and almost all existing cloud providers, *e.g.*, Microsoft, Amazon and VMWare, have been deploying software-based NFs on servers to maximize the flexibility [23, 30]. However, software NFs have two fundamental limitations – both stem from the nature of software packet processing. First, processing packets in software has limited capacity. Existing software NFs usually require multiple cores to achieve 10 Gbps rate [33, 43]. But the latest network links have scaled up to 40∼100 Gbps [11]. Although one could add more cores in a server, doing so adds significant cost, not only in terms of capital expense, but also more operational expense as they are burning significantly more energy. Second, processing packets in software incurs large, and highly variable latency. This latency may range from tens of microsecond to milliseconds [22,33,39]. For many low latency applications (*e.g.*, stock trading), this inflated latency is unacceptable.

NetBricks: Taking the V out of NFV

Aurojit Panda† Sangjin Han† Keon Jang‡ Melvin Walls† Sylvia Ratnasamy† Scott Shenker†*
† UC Berkeley ‡ Google * ICSI

Abstract

The move from hardware middleboxes to software network functions, as advocated by NFV, has proven more challenging than expected. Developing new NFs remains a tedious process, requiring that developers repeatedly rediscover and reapply the same set of optimizations, while current techniques for providing isolation between NFs (using VMs or containers) incur high performance overheads. In this paper we describe NetBricks, a new NFV framework that tackles both these problems. For building NFs we take inspiration from modern data analytics frameworks (*e.g.*, Spark and Dryad) and build a small set of customizable network processing elements. We also embrace type checking and safe runtimes to provide isolation in software, rather than rely on hardware isolation. NetBricks provides the same memory isolation as containers and VMs, without incurring the same performance penalties. To improve I/O

standard tools for managing VMs; (c) faster development, which now requires writing software that runs on commodity hardware; and (d) reduced costs by consolidating several NFs on a single machine. However, despite these promised advances, there has been little progress towards large-scale NF deployments. Our discussions with three major carriers revealed that they are only just beginning small scale test deployments (with 10-100s of customers) using simple NFs *e.g.*, firewalls and NATs.

The move from hardware middleboxes to software NFs was supposed to speed innovation, so why has progress been so slow? We believe this delay is because traditional approaches for both *building* and *running* NFs are a poor match for carrier networks, which have the following requirements: *performance*, NF deployments should be able to provide per-packet latencies on the order of 10s of μs, and throughput on the order of 10s of Gbps; *efficiency*, it should be possible to consolidate several NFs on a sin-

High performance NF implementations

Microboxes: High Performance NFV with Customizable, Asynchronous TCP Stacks and Dynamic Subscriptions

Guyue Liu*, Yuxin Ren*, Mykola Yurchenko*,
K.K. Ramakrishnan†, Timothy Wood*
*George Washington University, †University of California, Riverside

mOS: A Reusable Networking Stack for Flow Monitoring Middleboxes

Muhammad Jamshed, YoungGyoun Moon, Donghwi Kim, Dongsu Han, and KyoungSoo Park
School of Electrical Engineering, KAIST

FlowBlaze: Stateful Packet Processing in Hardware

Salvatore Pontarelli[1,2], Roberto Bifulco[3], Marco Bonola[1,2], Carmelo Cascone[4], Marco Spaziani[2,5], Valerio Bruschi[2,5], Davide Sanvito[6], Giuseppe Siracusano[3], Antonio Capone[6], Michio Honda[3], Felipe Huici[3] and Giuseppe Bianchi[2,5]

[1]Axbryd, [2]CNIT, [3]NEC Laboratories Europe, [4]Open Networking Foundation, [5]University of Rome Tor Vergata, [6]Politecnico di Milano

Abstract

While programma
handle growing ne
yet simple abstract
in hardware remai
problem with Flow
stateful packet pro
straction is based o
troduces the explic
Blaze to leverage
pressive, supportin
tions, and easy to u
tation issues from
FlowBlaze on a N
tency (in the order
tively little power,
thousands of flows
for even higher sp
ware and software
licly available.

1 Introductio

Network infrastruc
network functions
and server load ba
such as access co
examples. Given
the need to contin
tors have turned to

ClickNP: Highly Flexible and High Performance Network Processing with Reconfigurable Hardware

Bojie Li§† Kun Tan† Layong (Larry) Luo‡ Yanqing Peng•† Renqian Luo§† Ningyi Xu† Yongqiang Xiong† Peng Cheng† Enhong Chen§
†Microsoft Research §USTC ‡Microsoft •SJTU

RACT

exible software network functions (NFs) are cru-
ponents to enable multi-tenancy in the clouds. How-
ware packet processing on a commodity server has
apacity and induces high latency. While software
d scale out using more servers, doing so adds sig-
ost. This paper focuses on accelerating NFs with
nable hardware, *i.e.*, FPGA, which is now a ma-
nology and inexpensive for datacenters. However,
predominately programmed using low-level hard-
cription languages (HDLs), which are hard to code
ult to debug. More importantly, HDLs are almost
ble for most software programmers. This paper presents
a FPGA-accelerated platform for highly flexible
performance NFs with commodity servers. ClickNP
flexible as it is completely programmable using
l C-like languages, and exposes a modular program-
raction that resembles Click Modular Router. ClickNP
gh performance. Our prototype NFs show that they
ss traffic at up to 200 million packets per second
-low latency (< 2μs). Compared to existing soft-
nterparts, with FPGA, ClickNP improves through-
0x, while reducing latency by 10x. To the best of
ledge, ClickNP is the first FPGA-accelerated plat-
NFs, written completely in high-level language and
40 Gbps line rate at any packet size.

oncepts

1. INTRODUCTION

Modern multi-tenant datacenters provide shared infrastructure for hosting many different types of services from different customers (*i.e.*, tenants) at a low cost. To ensure security and performance isolation, each tenant is deployed in a *virtualized network* environment. Flexible network functions (NFs) are required for datacenter operators to enforce isolation while simultaneously guaranteeing Service Level Agreements (SLAs).

Conventional hardware-based network appliances are not flexible, and almost all existing cloud providers, *e.g.*, Microsoft, Amazon and VMWare, have been deploying software-based NFs on servers to maximize the flexibility [23, 30]. However, software NFs have two fundamental limitations – both stem from the nature of software packet processing. First, processing packets in software has limited capacity. Existing software NFs usually require multiple cores to achieve 10 Gbps rate [33, 43]. But the latest network links have scaled up to 40∼100 Gbps [11]. Although one could add more cores in a server, doing so adds significant cost, not only in terms of capital expense, but also more operational expense as they are burning significantly more energy. Second, processing packets in software incurs large, and highly variable latency. This latency may range from tens of microsecond to milliseconds [22,33,39]. For many low latency applications (*e.g.*, stock trading), this inflated latency is unacceptable.

NetBricks: Taking the V out of NFV

Aurojit Panda† Sangjin Han† Keon Jang‡ Melvin Walls† Sylvia Ratnasamy† Scott Shenker†*
† UC Berkeley ‡ Google * ICSI

Abstract

The move from hardware middleboxes to software network functions, as advocated by NFV, has proven more challenging than expected. Developing new NFs remains a tedious process, requiring that developers repeatedly rediscover and reapply the same set of optimizations, while current techniques for providing isolation between NFs (using VMs or containers) incur high performance overheads. In this paper we describe NetBricks, a new NFV framework that tackles both these problems. For building NFs we take inspiration from modern data analytics frameworks (*e.g.*, Spark and Dryad) and build a small set of customizable network processing elements. We also embrace type checking and safe runtimes to provide isolation in software, rather than rely on hardware isolation. NetBricks provides the same memory isolation as containers and VMs, without incurring the same performance penalties. To improve I/O

standard tools for managing VMs; (c) faster development, which now requires writing software that runs on commodity hardware; and (d) reduced costs by consolidating several NFs on a single machine. However, despite these promised advances, there has been little progress towards large-scale NF deployments. Our discussions with three major carriers revealed that they are only just beginning small scale test deployments (with 10-100s of customers) using simple NFs *e.g.*, firewalls and NATs.

The move from hardware middleboxes to software NFs was supposed to speed innovation, so why has progress been so slow? We believe this delay is because traditional approaches for both *building* and *running* NFs are a poor match for carrier networks, which have the following requirements: *performance*, NF deployments should be able to provide per-packet latencies on the order of 10s of μs, and throughput on the order of 10s of Gbps; *efficiency*, it should be possible to consolidate several NFs on a sin-

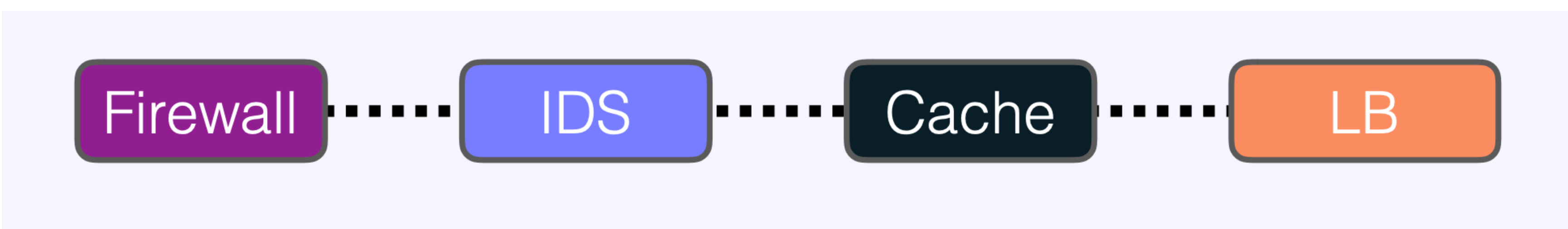
NetBricks: Taking the V out of NFV

OSDI'16

Slides borrowed from the OSDI talk

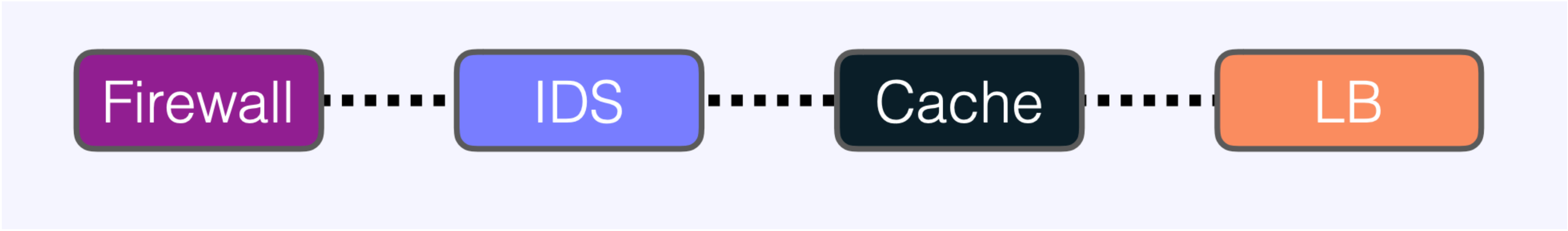
NFV Requirements

- **High Packet Rates:** Must keep up with line rate which is >10MPPS
- **Low Latency:** Used for applications like VoIP and video conferencing
- **Support NF Chaining:** Packets go through sequence of NFs



NFV Requirements

- **High Packet Rates:** Must keep up with line rate which is >10MPPS
- **Low Latency:** Used for applications like VoIP and video conferencing
- **Support NF Chaining:** Packets go through sequence of NFs



Challenges for NFV

- **Running NFs:**
  - Isolation and Performance
- **Building NFs:**
  - High-level Programming and Performance

Challenges for NFV

- Running NFs:
  - Isolation and Performance
- Building NFs:
  - High-level Programming and Performance

Isolation

- **Memory Isolation:** Each NF's memory cannot be accessed by other NFs.

- **Packet Isolation:** When chained, each NF processes packets in isolation.

- **Performance Isolation:** One NF does not affect another's performance.

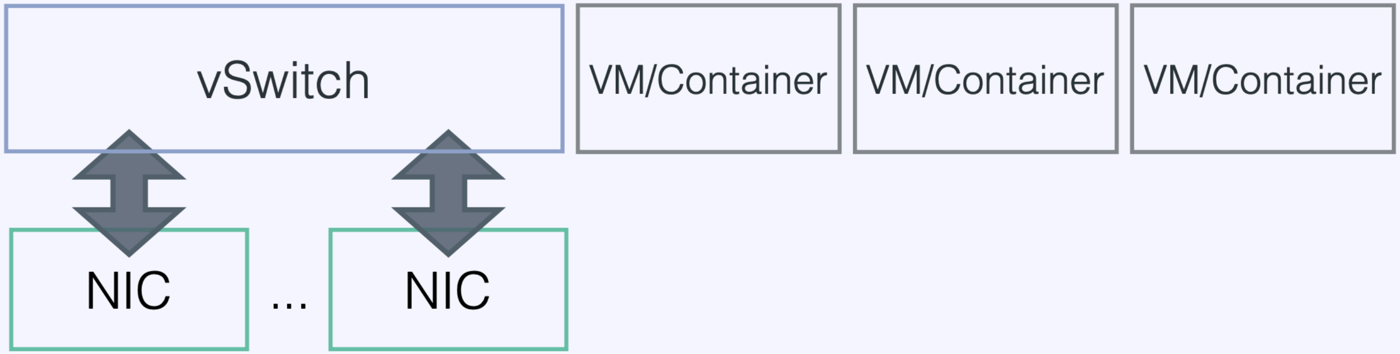
Isolation

- **Memory Isolation:** Each NF's memory cannot be accessed by other NFs.

- **Packet Isolation:** When chained, each NF processes packets in isolation.

- ~~**Performance Isolation:** One NF does not affect another's performance.~~

Current Solution

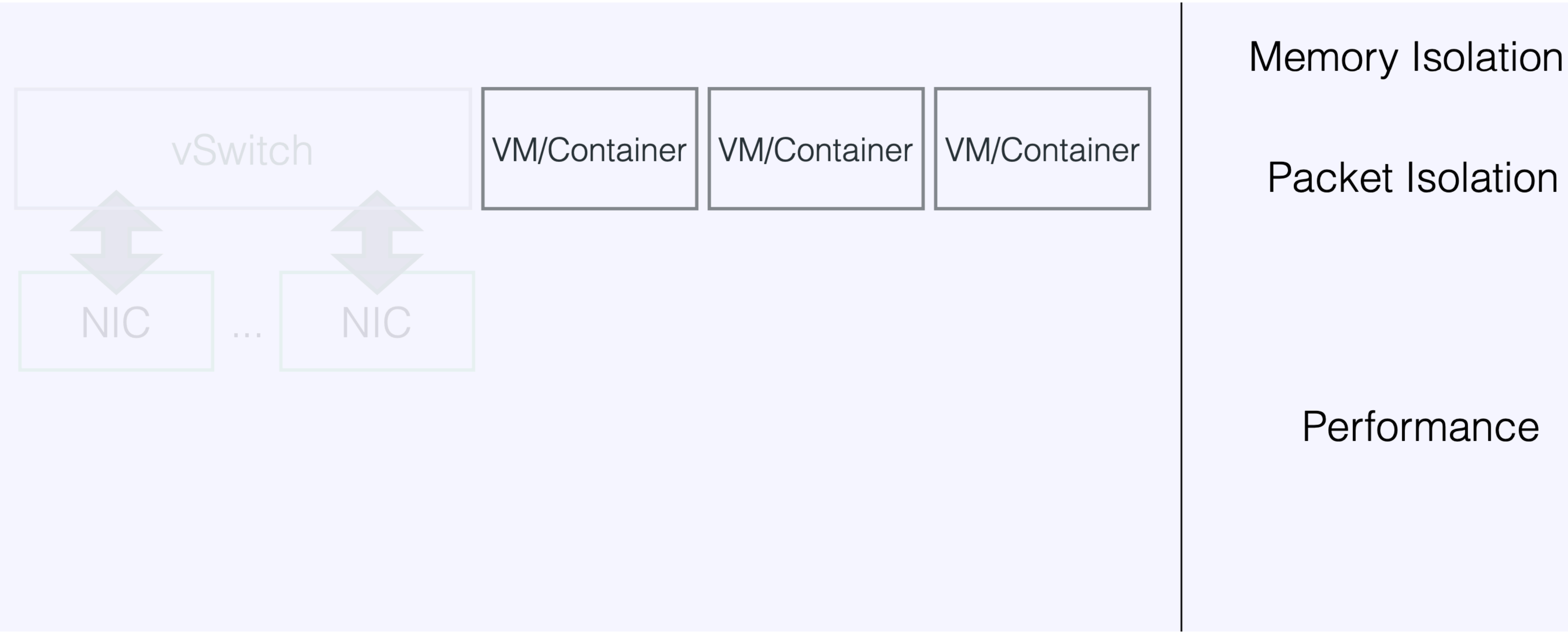


Memory Isolation

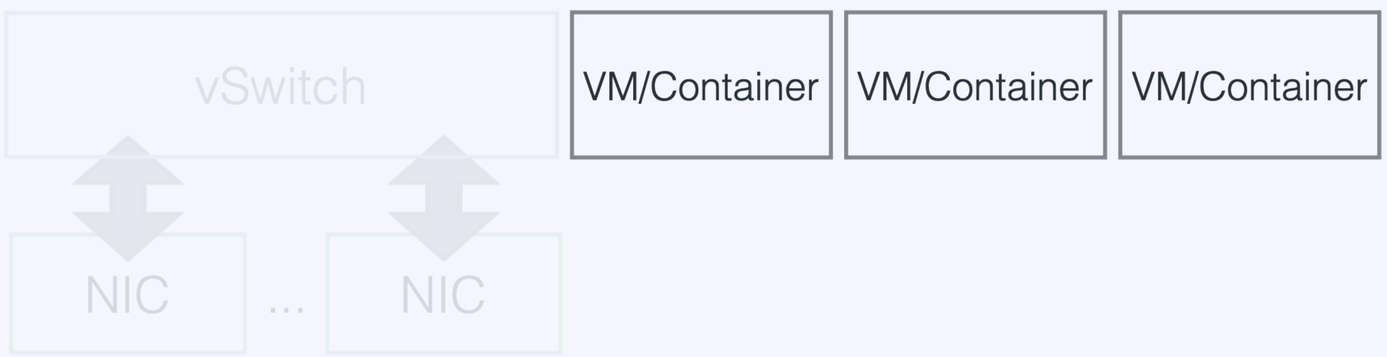
Packet Isolation

Performance

Current Solution



Current Solution

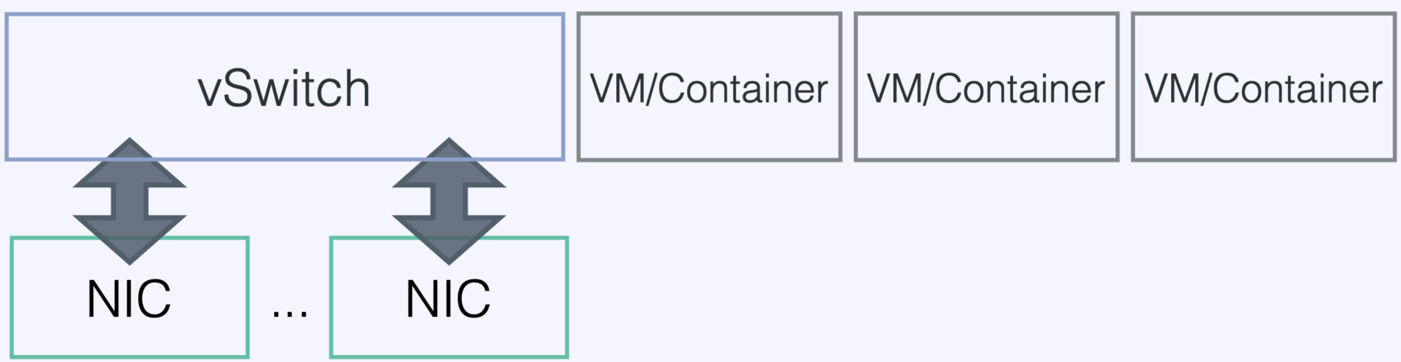


✔ Memory Isolation

Packet Isolation

Performance

Current Solution

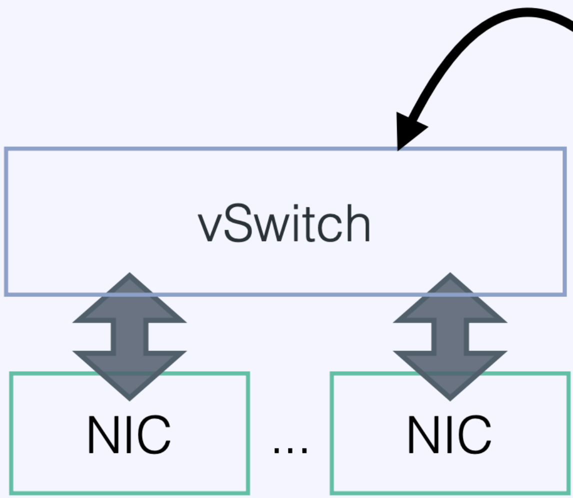


✔Memory Isolation

Packet Isolation

Performance

Current Solution

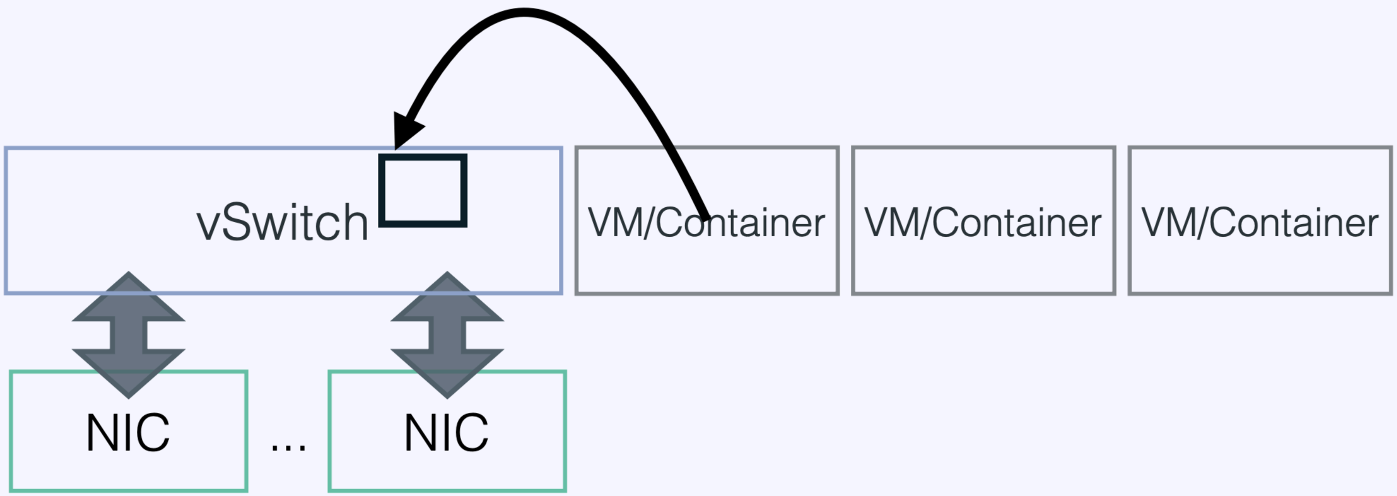


✔ Memory Isolation

Packet Isolation

Performance

Current Solution

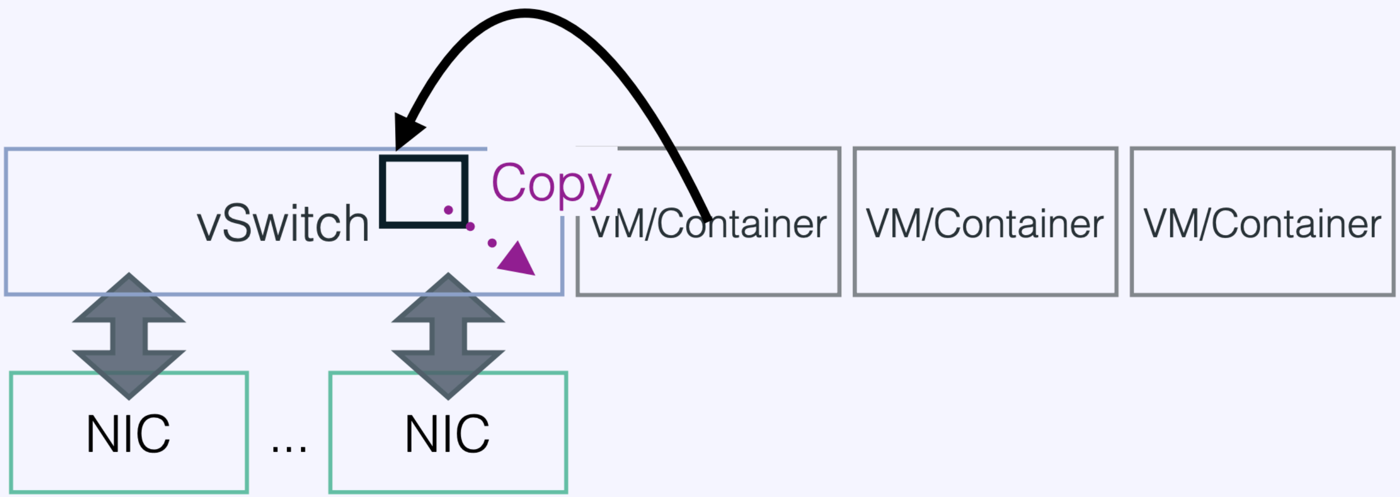


✔ Memory Isolation

Packet Isolation

Performance

Current Solution

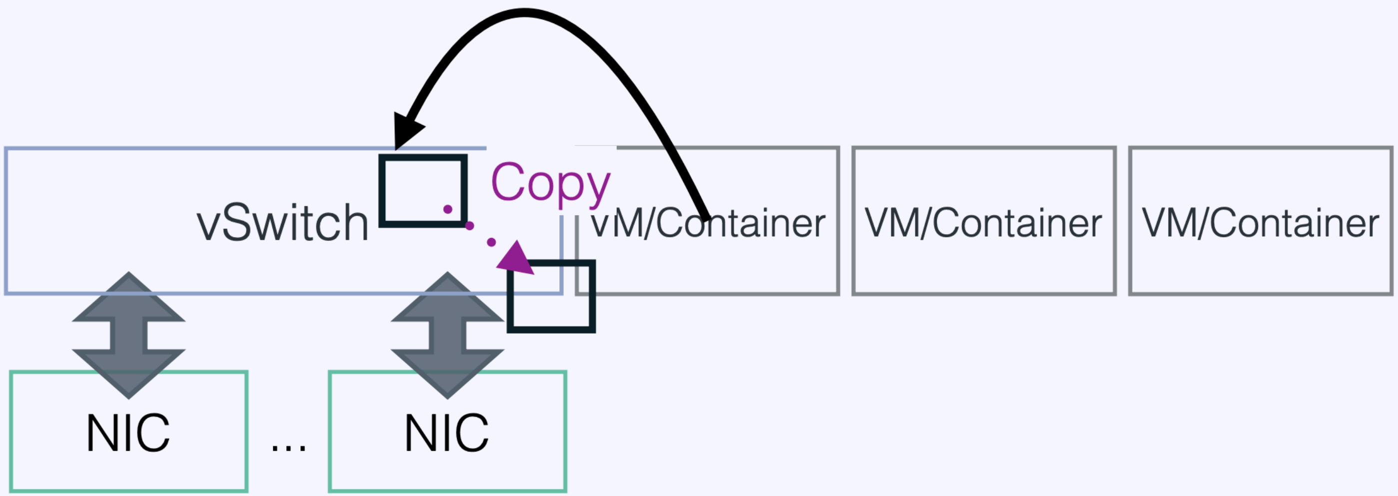


✔ Memory Isolation

Packet Isolation

Performance

Current Solution

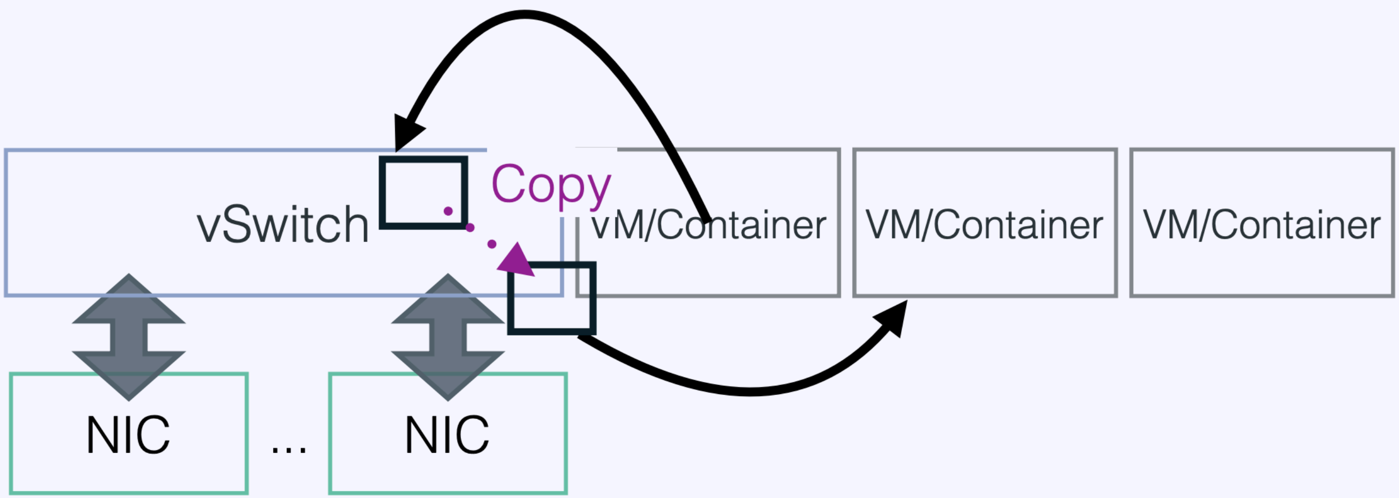


✔ Memory Isolation

Packet Isolation

Performance

Current Solution

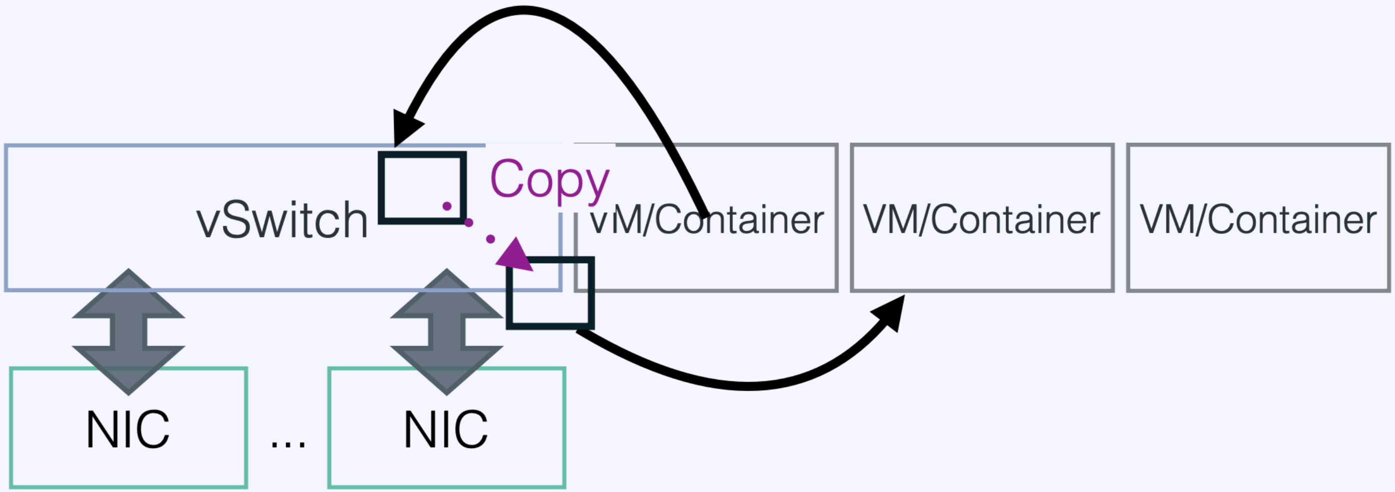


✔Memory Isolation

Packet Isolation

Performance

Current Solution

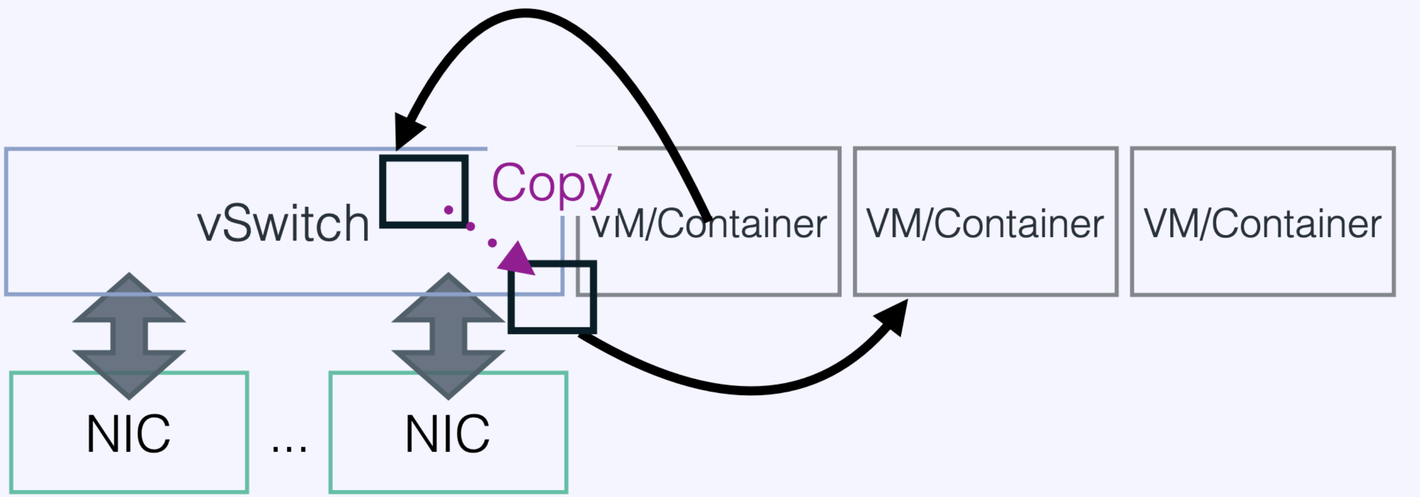


✔ Memory Isolation

✔ Packet Isolation

Performance

Current Solution

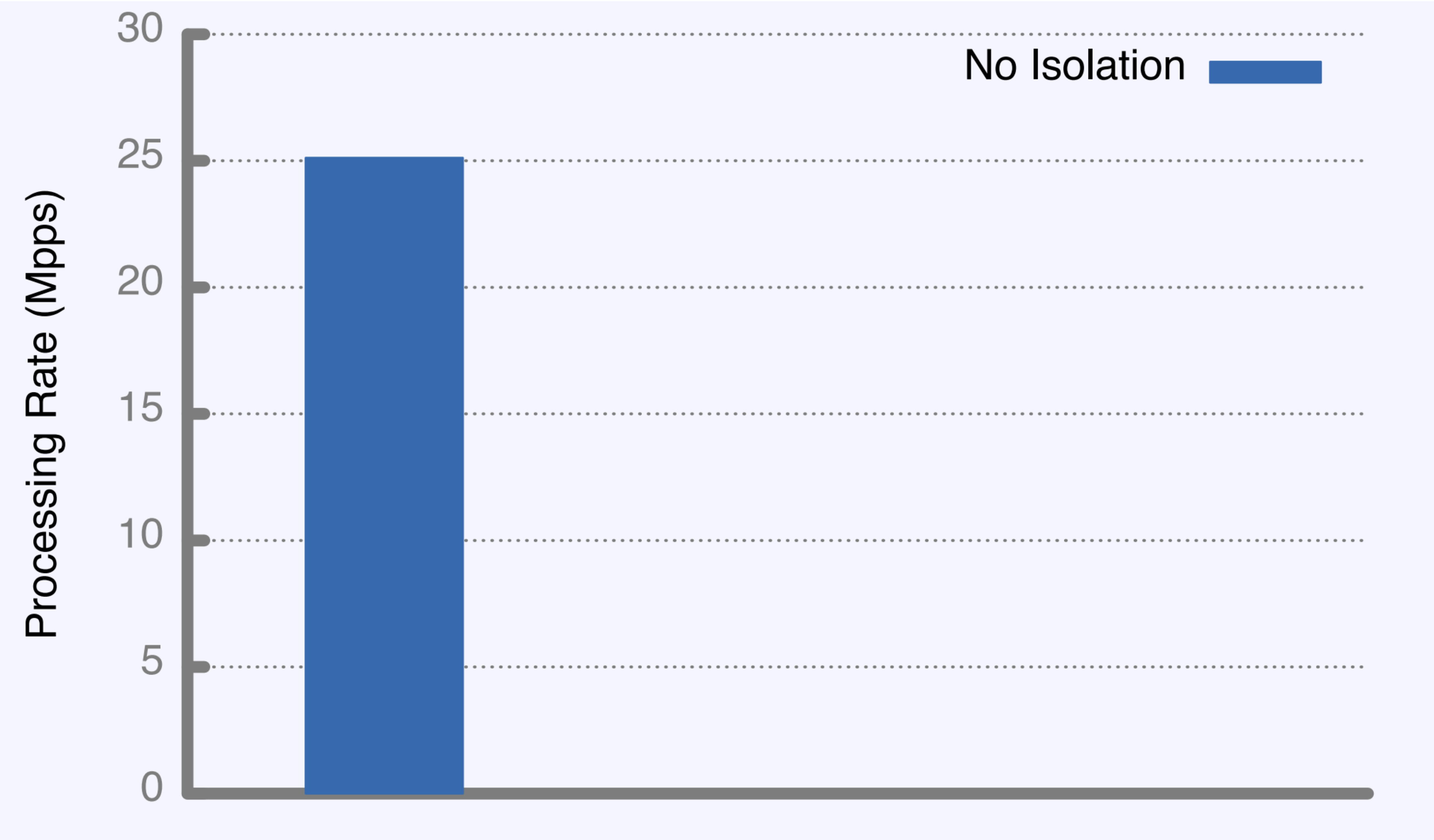


✔ Memory Isolation

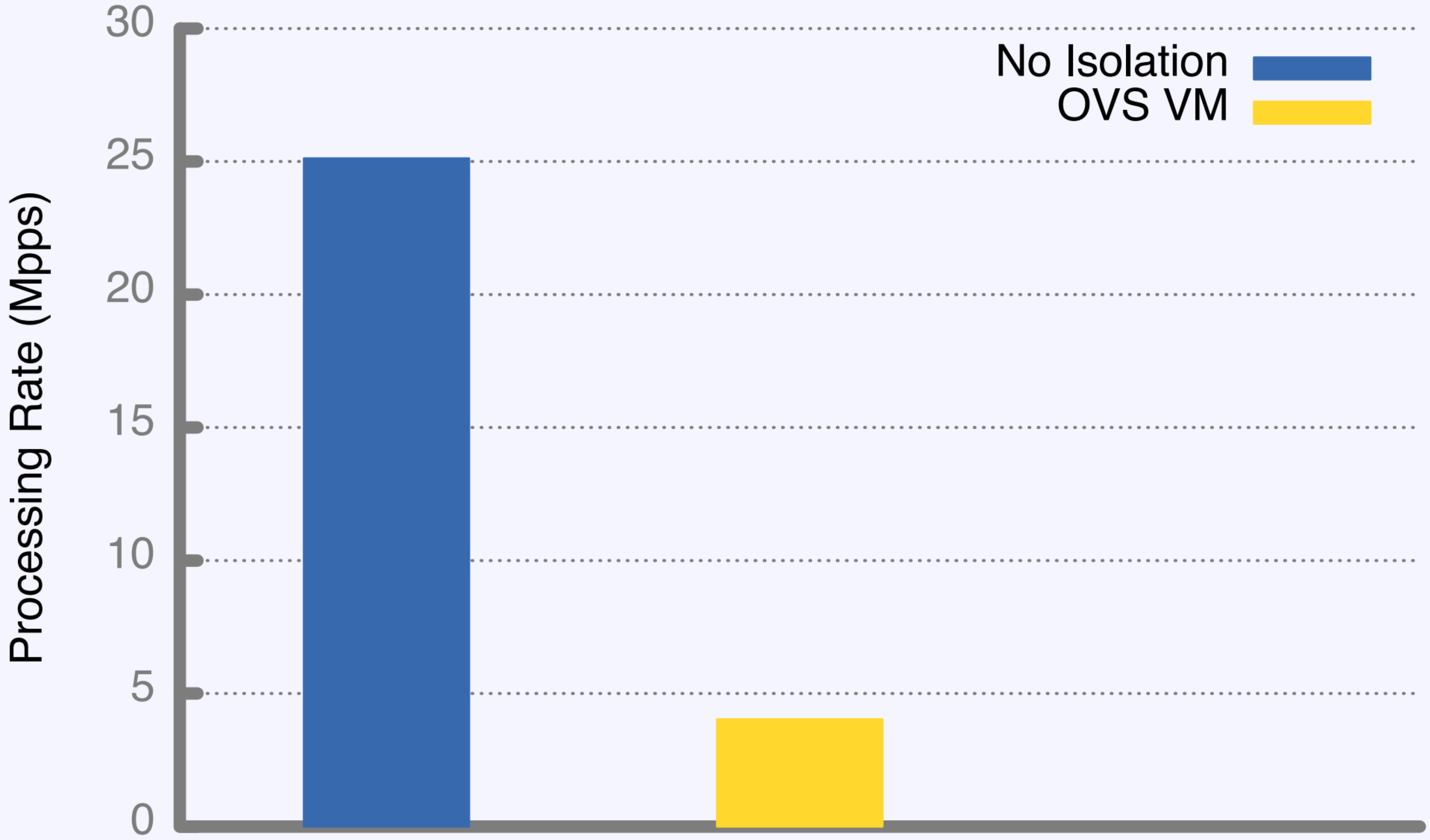
✔ Packet Isolation

✗ Performance

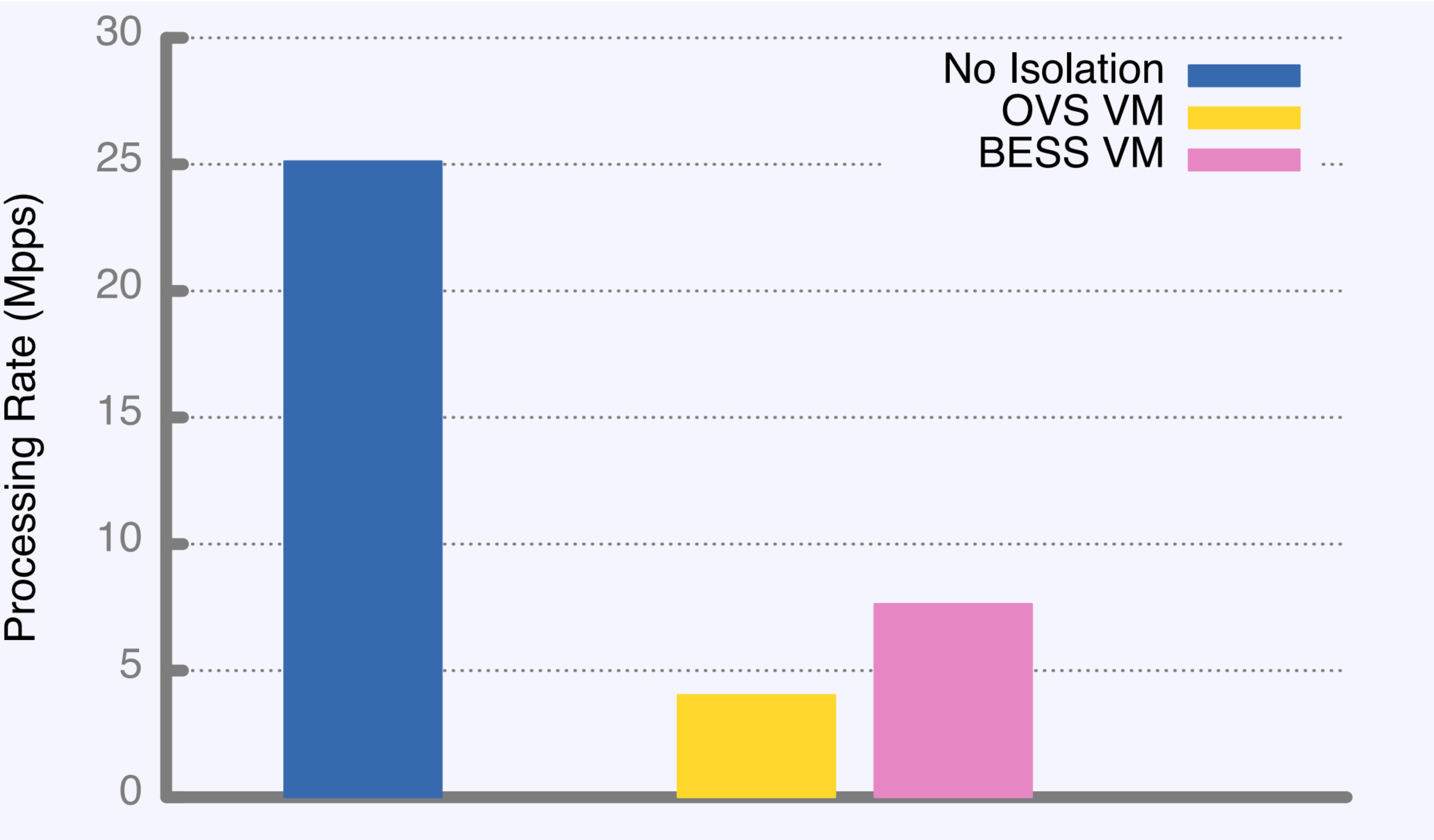
Isolation costs Performance



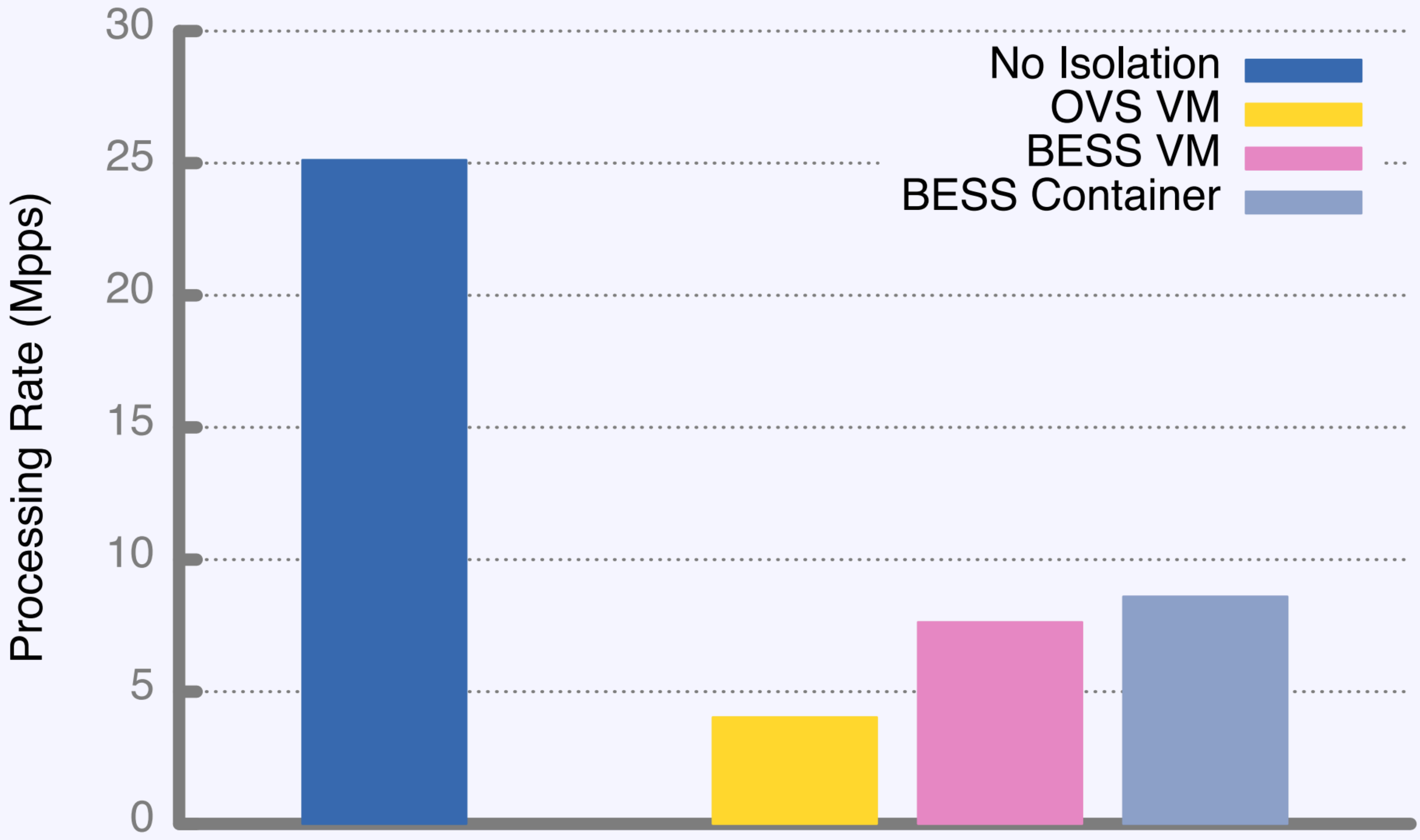
Isolation costs Performance



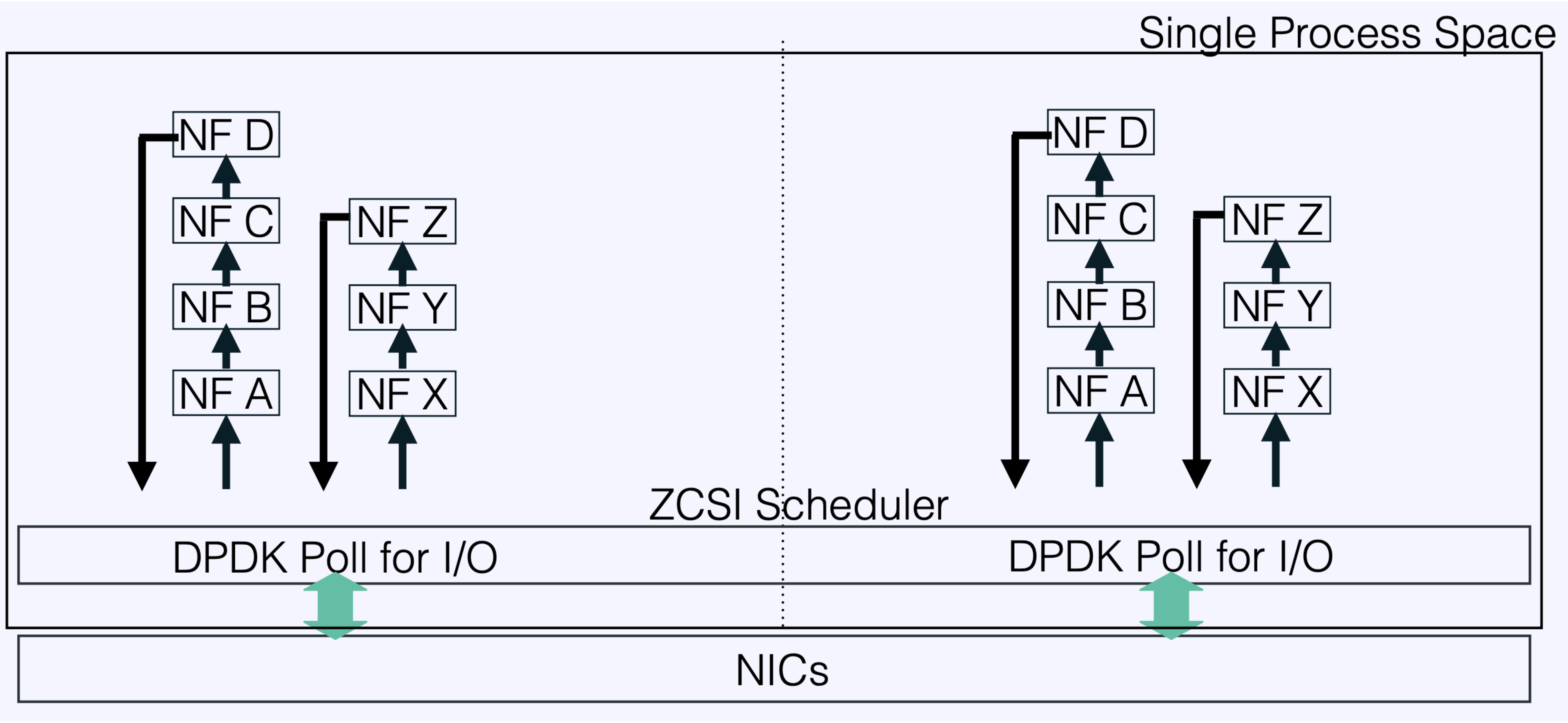
Isolation costs Performance



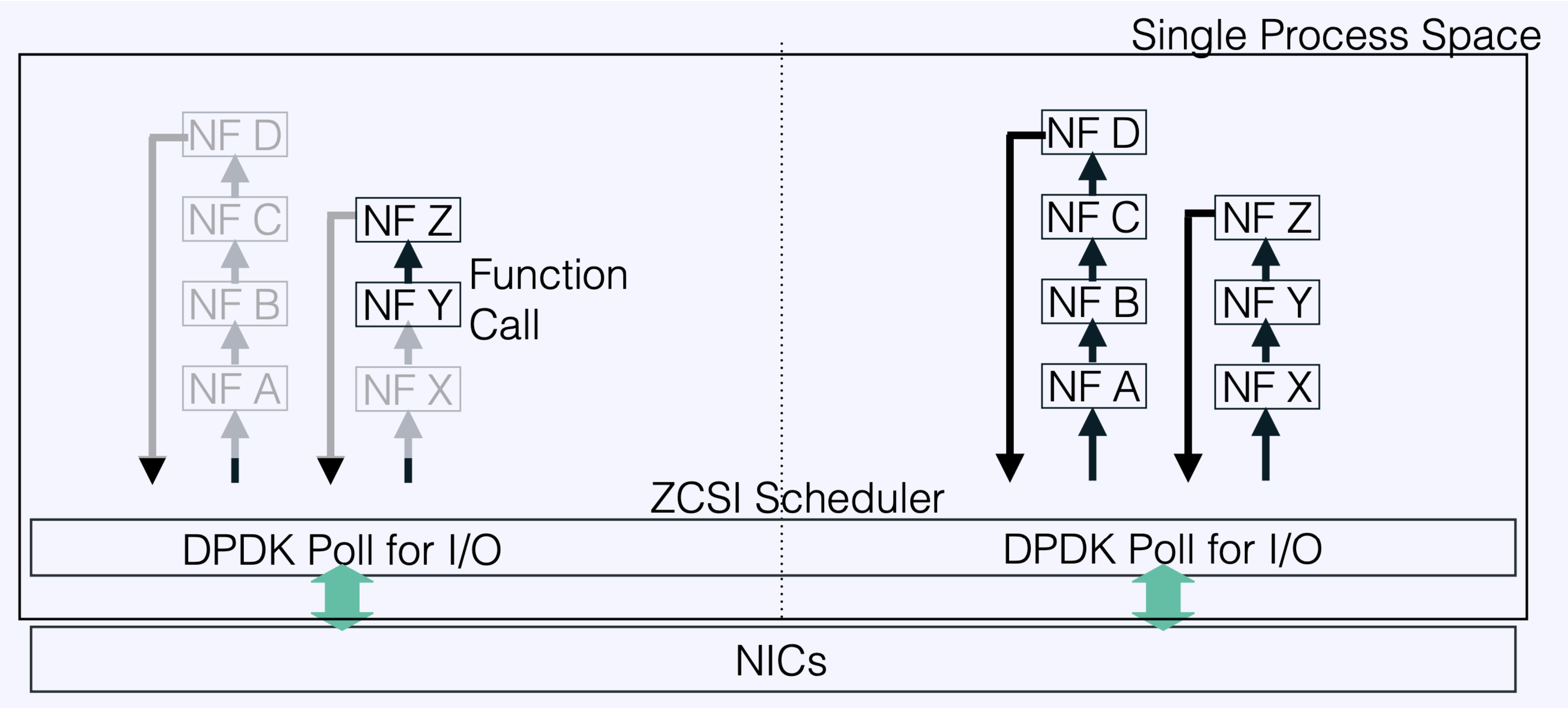
Isolation costs Performance



NetBricks Runtime Architecture



NetBricks Runtime Architecture



ZCSI: Zero Copy Soft Isolation

- VMs and containers impose cost on packets crossing isolation boundaries.
- Insight: Use type checking (compile time) and runtime checks for isolation.
- Isolation costs largely paid at compile time (small runtime costs).

NetBricks Approach

- Disallow pointer arithmetic in NF code: use safe subset of languages.
- Type checks + array bounds checking provide memory isolation.
- Build on unique types for packet isolation.
  - Unique types ensure references destroyed after certain calls.
  - Ensure only one NF has a reference to a packet.
  - Enables zero copy packet I/O.
- All of these features implemented on top of **Rust**.

Software Isolation

- Provides memory and packet isolation.
- Improved consolidation: multiple NFs can share a core.
  - Function call to NF (~ few cycles) vs context switch (~1 μs).
- Reduce memory and cache pressure.
  - Zero copy I/O => do not need to copy packets around.

Challenges for NFV

- Running NFs:
  - Isolation and Performance
- Building NFs:
  - High-level Programming and Performance

How to write NFs?

- **Current:** NF writers concerned about meeting performance targets
  - Low level abstractions (I/O, cache aware data structures) and low level code.
- Spend lots of time optimizing how abstractions are used to get performance.
- **Observation:** NFs exhibit common patterns: abstract and optimize these.
- Analogous to what happened in other areas.
  - MPI to Map Reduce, etc.

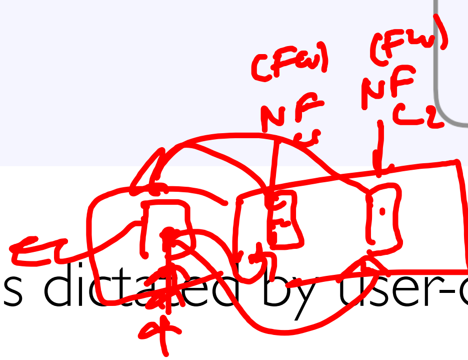
Abstractions

Packet Processing
- Parse/Deparse
- Transform
- Filter

Control Flow
- Group By
- Shuffle
- Merge

Byte Stream
- Window
- Packetize

State
- Bounded
- Consistency



Behavior of these abstractions dictated by user-defined functions (UDFs)

Example NF

- Maglev: Load balancer from Google (NSDI'16).
- NetBricks implementation: 105 lines, 2 hours of time.
- Comparable performance to optimized code

Conclusion

- Software isolation is necessary for high performance NFV.
  - Type checking + bound checking + unique types.
- Performance is not anathema to high-level programming
  - Abstract operators + UDF simplify development.

Your Opinions

- Pros
  - UDFs provide developers with flexibility and operators with high performance.
  - Reduce overhead for memory/packet isolation
    - Moves away from using container and VMs
    - Software memory isolation with compile-time and runtime checks.
    - Found value for Rust's memory-safe programming for networking.
  - Thorough evaluation, high performance.
  - Provides clean, easy-to-use primitives.

Your Opinions

- Cons
  - Clean-slate: requires rewriting NFs
  - Potentially high CPU utilization.
  - Can VMs and containers provide stronger isolation guarantees (e.g. performance isolation)? What if there are bugs in NF implementation?
  - How to achieve line-rate performance for very complex NFs?

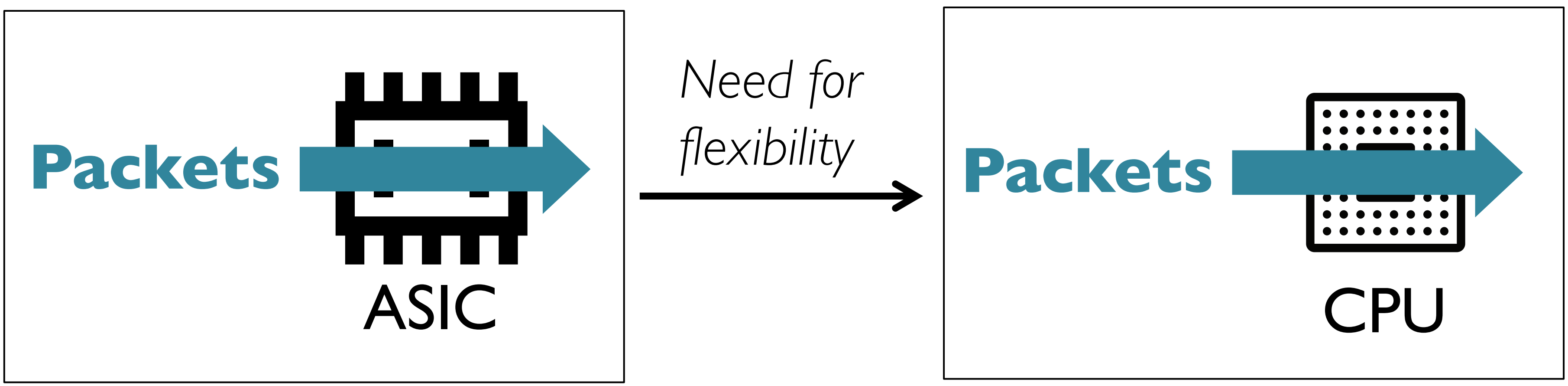
Your Opinions

- Ideas
  - Intel SGX to provide greater security and isolation?
  - In-depth evaluation of security of NetBricks.
  - Improving the programming interfaces.
  - Using programmable hardware for NFs

Evolution of middleboxes

Dedicated hardware

Software

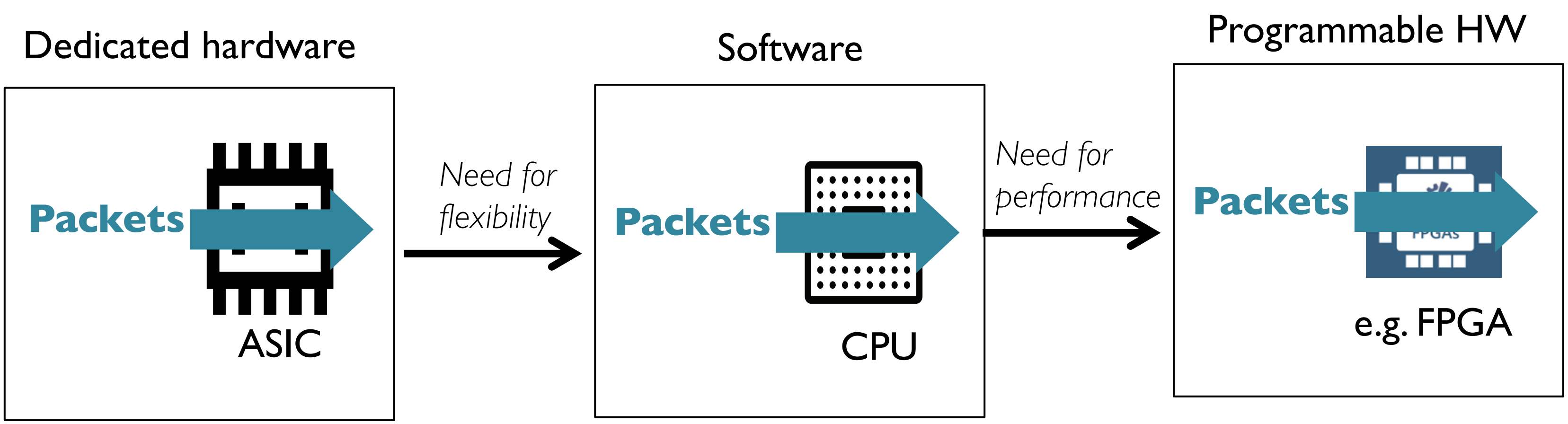


Middleboxes

Network functions

contents of the slide borrowed from talks given by Aurojit Panda, NYU

Evolution of middleboxes

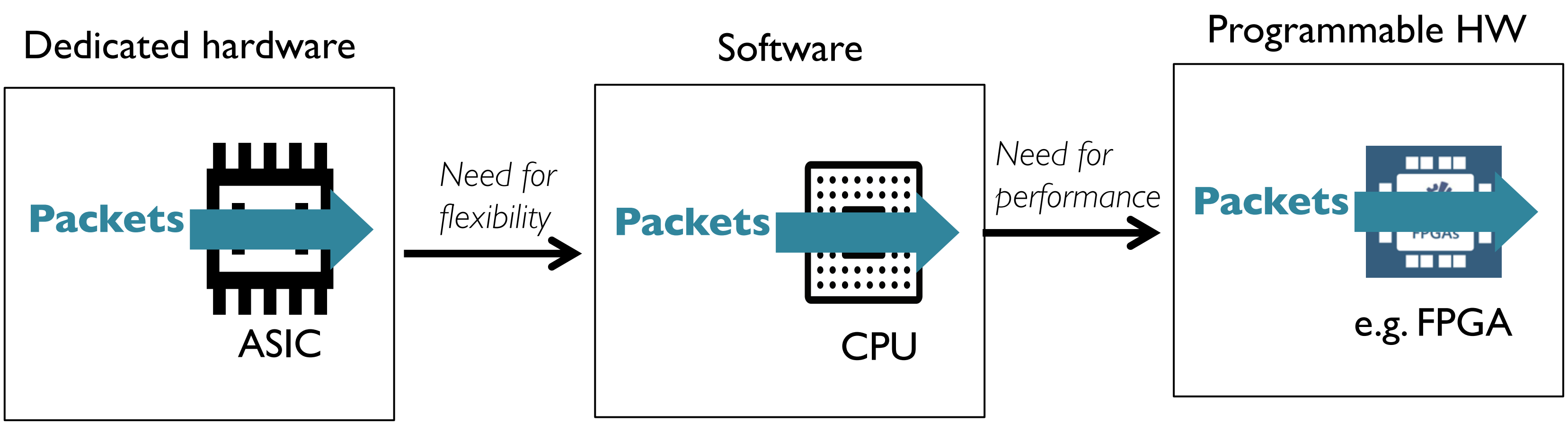


Middleboxes

Network functions

ClickNP, SIGCOMM'16
FlowBlaze, NSDI'19
TEA, SIGCOMM'20

Evolution of middleboxes



Middleboxes

Network functions

ClickNP, SIGCOMM'16
FlowBlaze, NSDI'19
TEA, SIGCOMM'20

Logistics

- Tuesday, Dec 1st: Students' presentation (and choice)
  - Sign up for a paper by the end of this week.
- Thursday, Dec 3rd: No reading assignment, only wrap-up lecture.
- Friday, Dec 4th: Final project report due.
- Tuesday, Dec 8th: Project presentation. Details TBA.