Network Functions

ECE/CS598HPN

Radhika Mittal
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*
Conventional view of networks

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.

*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*
Rise of middleboxes

Stringent performance requirements: process packets at line rate with minimal latency overhead.

*Contents of the slide borrowed from talks given by Aurojit Panda, NYU*
Evolution of middleboxes

Dedicated hardware

Packets
ASIC

Middleboxes

Need for flexibility

Software

Packets
CPU

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 Diagram](image_url)
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

Elastic Scaling of Stateful Network Functions

Shinuo Wen1, Justine Sherry2, Sangjin Han3, Sue Moon4, Sylvia Ratnasamy5, and Scott Shenker6
1University of California, Berkeley 2KAIST 3CMU 4ICS

Abstract

Elastic scaling is a central problem hard to realize in practice. The most Network Functions (NFs) need to be shared across NFV nodes while meeting requirements placed on NFs. No solution exists that meets the performance goals for the full spectrum of NFs. S6 is a new framework of NFs without compromising builds on the insight that abstraction is well-suited to the setting as a distributed shared

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

Murad Kablan, Azzam Alsadah, Eric Keller
University of Colorado, Boulder

Pico Replication: A High Availability Framework for Middleboxes

Shriram Rajagopalan1, Dan Williams1, Hani Jamjoom1
1IBM T. J. Watson Research Center, Yorktown Heights, NY

Abstract

Middleboxes are being engineered, compiled, and deployed at scale. However, the problem of persisting state is plaguing NFV deployments of shared middleboxes. We present PicoReplication (PR), a framework that exploits a hash table to achieve low overhead, high availability, and strong guarantees.

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

Anat Bremmer-Bart1, Scott Shenker1
1School of Computer Science, Carnegie Mellon University

Abstract

OpenBox is a software-defined framework that enables developers to develop and deploy software-defined middleboxes on commodity hardware.

E2: A Framework for NFV Applications

Shoumik Palkar UC Berkeley
spalkar@berkeley.edu

Chang Lan UC Berkeley
clan@eecs.berkeley.edu

Keon Jang Intel Labs
keon.jang@intel.com

Sangjin Han UC Berkeley
sangjin@eecs.berkeley.edu

Aurojit Panda UC Berkeley
apanda@cs.berkeley.edu

Luigi Rizzo Università di Pisa
rizzo@iet.unipi.it

Scott Shenker UC Berkeley and ICSI
shenker@icsi.berkeley.edu

Shiriram Rajagopalan1, Dan Williams1, Hani Jamjoom1
1IBM T. J. Watson Research Center, Yorktown Heights, NY

Abstract

Network functions; Middleboxes; Software-Defined Networks; NFV

1. INTRODUCTION

In solving important problems in the forwarding plane, such as cost, management, multi-tenancy, and high efficiency, middleboxes could not be easily adapted to other classes. Unlike generic cloud applications, middleboxes share a commonality in the way they are deployed and managed. This includes the need for high availability and resilience, as well as the ability to deploy new network functionality as add-on modules.

To rectify this situation, network operators are moving to a state-centric, systems-level abstraction for evaluating middleboxes. This involves partitioning the network into smaller, isolated contexts of execution [22, 26, 31]. Middleboxes are defined in terms of their behavior, and their code is annotated to perform custom state allocation, their code to perform custom state allocation, and state objects that need explicit handling. Using state-affecting algorithms such as parallel release and ordered logging, recovery is always correct.

To address the challenges of managing middleboxes, we propose S6, a development and runtime framework that supports stateful virtualization of middleboxes. S6 abstracts the physical network to provide a unified, isolated context of execution. Middleboxes are defined in terms of their behavior, and their code is annotated to perform custom state allocation, state objects that need explicit handling. Using state-affecting algorithms such as parallel release and ordered logging, recovery is always correct.

Abstract

Middleboxes are being engineered, compiled, and deployed at scale. However, the problem of persisting state is plaguing NFV deployments of shared middleboxes. We propose PicoReplication (PR), a framework that exploits a hash table to achieve low overhead, high availability, and strong guarantees.

OpenBox is a software-defined framework that enables developers to develop and deploy software-defined middleboxes on commodity hardware.

E2 is a framework for NFV applications that allows developers to easily create, manage, and deploy middleboxes on commodity hardware.

Paving the way for NFV:
Simplifying Middlebox Modifications using StateAlyzr

Junaid Khalid, Aaron Gemmer-Jacobson, Roney Michael, Anubhavindhu Abhashkumar, Aditya Akella
University of Wisconsin-Madison

Abstract

StateAlyzr is a novel, framework-based development and runtime framework that enables developers to easily create, manage, and deploy middleboxes on commodity hardware.
Understanding NF Performance

Backtracking Algorithmic Complexity Attacks Against a NIDS
Randy Smith, Cristian Estan, Sontosh Jha
Computer Sciences Department

Automated Synthesis of Adversarial Workloads for Network Functions
Luis Pedroso, EPFL
luis.pedroso@epfl.ch
Rishabh Iyer, EPFL
rishabh.iyer@epfl.ch
Arseny Zaostrovnykh, EPFL
arseny.zaostrovnykh@epfl.ch
Jonas Fietz, EPFL
jonas.fietz@epfl.ch
Katerina Argyraki, EPFL
katerina.argyraki@epfl.ch

Denial of Service via Algorithmic Complexity Attacks
Scott A. Crosby
scotscro@cs.ri.edu
Department of Computer Science, Rice University

PerfSight: Performance Diagnosis for Software Dataplanes
Wenlei Wu, Keqiang He, Aditya Akella
University of Wisconsin-Madison

Making DPI Engines Resilient to Algorithmic Complexity Attacks
Yehuda Afek, Member, IEEE, Anat Bremer-Bart, Member, IEEE, Yotam Harchol, Member, IEEE, David Hay, Member, IEEE, and Yaron Koren, Member, IEEE

NFPerf: Online Performance Monitoring and Bottleneck Detection for NFV
Priyanka Naik, Dilip Kumar Shaw, Mythili Vutukuru
Department of Computer Science and Engineering, Indian Institute of Technology, Bombay
Email: {priyanka, dilipk3, mythili}@iitb.ac.in

Abstract
Automated Synthesis of Adversarial Workloads for Network Functions

While balanced tree algorithms, such as red-black and AVL trees [11], AVL trees [11], and treaps [11] can be used to store and efficiently manipulate key-value pairs, they are not always the best choice for implementing hash tables. In the worst case, these algorithms can suffer from high overhead due to the need for frequent rebalancing operations. Moreover, while hash tables are widely used in many applications, they are not always the most efficient choice for storing and retrieving key-value pairs. The worst-case time complexity of hash table operations is O(n) for insertion, deletion, and lookup operations, which can lead to performance bottlenecks in high-traffic scenarios. In this paper, we present a new class of low-bandwidth denial of service attacks that exploit algorithmic deficiencies in hash table implementations. These attacks can cause hash table implementations to become unstable, leading to high overhead and performance degradation.

Denial of Service via Algorithmic Complexity Attacks

We present a new class of low-bandwidth denial of service attacks that exploit algorithmic deficiencies in many common applications' data structures. Frequently used data structures have “average-case” expected running time that’s far more efficient than the worst-case. For example, both binary trees and hash tables can degenerate to linked lists with care. We demonstrate that attackers can force legitimate users to perform O(n) operations to insert n elements. Furthermore, we present a framework for generating adversarial workloads that can cause hash table implementations to become unstable. Our attacks are effective against both naive and optimized hash table implementations.

PerfSight: Performance Diagnosis for Software Dataplanes

Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions using FlowTags
Seyed Kaveh Fayazbaksh*, Luis Chiang†, Vyas Sekar*, Minlan Yu†, Jeffrey C. Mogul∗
∗Carnegie Mellon University †Deutsche Telekom Labs

Abstract
Middleboxes provide key security and performance guarantees in networks. Unfortunately, the dynamic traffic modifications these devices make it difficult to reason about network-wide policy enforcement. In this paper, we present PerfSight, a novel flow-based monitoring system that can provide performance diagnosis and network-wide policy enforcement. Our system allows network operators to monitor and control traffic flows across a network in real-time.

Making DPI Engines Resilient to Algorithmic Complexity Attacks

Yehuda Afek, Member, IEEE, Anat Bremer-Bart, Member, IEEE, Yotam Harchol, Member, IEEE, David Hay, Member, IEEE, and Yaron Koren, Member, IEEE

Abstract
DPI engines are used to detect and analyze network traffic, but they can be vulnerable to algorithmic complexity attacks. In this paper, we present a new class of low-bandwidth denial of service attacks that exploit algorithmic deficiencies in hash table implementations. These attacks can cause DPI engines to become unstable, leading to high overhead and performance degradation.
Providing guarantees about NF behavior

Software Dataplane Verification

A Formally Verified NAT

SymNet: scalable symbolic execution for modern networks
High performance NF implementations

Microboxes: High Performance NFV with Customizable, Asynchronous TCP Stacks and Dynamic Subscriptions
Guyue Liu1, Yuxin Ren1, Mykola Yurchenkov2, K.K. Ramakrishnan1, Timothy Wood1
1*George Washington University, 2University of California, Riverside

FlowBlaze: Stateful Packet Processing in Hardware
Salvatore Pontarelli1,2, Roberto Bifulco1, Marco Bonola1,2, Carmelo Cascone1, Marco Spaziani1,2, Valerio Bruschi1,2, Davide Sanvitto1,2, Giuseppe Siracusano1,2, Antonio Capone1, Michele Honda1, Filipe Hucil1 and Giuseppe Bianchi1,2
1Axbryd, 2CNIT, 3NCE Laboratories Europe, 4Open Networking Foundation, 5University of Rome Tor Vergata, 6Politecnico di Milano

Abstract
While programmable middleboxes remain a popular middlebox type–from NFs requiring only this notice and the full citation on the made or distributed for pro
tor commercial advantage and that copies bear

NetBricks: Taking the V out of NFV
Aurojit Panda1, Sangjin Han1, Keon Jang1, Melvin Walls1, Sylvia Ratnasamy1, Scott Shenker1
1 UC Berkeley 2 Google 3 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 DO

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

Middleboxes: High Performance NFV with Customizable, Asynchronous TCP Stacks and Dynamic Subscriptions
Guyue Liu1, Yuxin Ren1, Mykola Yurchenkov2, K.K. Ramakrishnan1, Timothy Wood1
1*George Washington University, 2University of California, Riverside

1 INTRODUCTION
Network infrastructure network functions (and server load balancers) such as access control examples. Given the need to continue to be effective in a world where services have turned to

RACT
Flexible software network functions (NFs) are crucial to enable multi-tenant tenancy in the clouds. How

1. INTRODUCTION
Modern multi-tenant datacenter providers share 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 seen 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 can range from tens of microseconds to milliseconds [22, 33, 39]. For many low latency applications (e.g., stock trading), this inflated latency is unacceptable.

mOS: A Reusable Networking Stack for Flow Monitoring Middleboxes
Muhammad Jamiolahmed, YoungGyou Moon, Dongwi Kim, Dongsoo Han, and KyoungSoo Park
School of Electrical Engineering, KAIST

Abstract
This work presents the design and implementation of mOS: A Reusable Networking Stack for Flow Monitoring Middleboxes. In this paper we propose a novel networking stack that can be used for flow monitoring middleboxes. mOS is designed to be flexible and efficient, allowing for easy customization of the stack’s functionality and providing a high performance networking stack. Our evaluation demonstrates that mOS outperforms existing middlebox stacks by several orders of magnitude while providing a simple and flexible way to implement flow monitoring middleboxes. This work presents the design and implementation of mOS: A Reusable Networking Stack for Flow Monitoring Middleboxes. In this paper we propose a novel networking stack that can be used for flow monitoring middleboxes. mOS is designed to be flexible and efficient, allowing for easy customization of the stack’s functionality and providing a high performance networking stack. Our evaluation demonstrates that mOS outperforms existing middlebox stacks by several orders of magnitude while providing a simple and flexible way to implement flow monitoring middleboxes.
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

FlowBlaze: Stateful Packet Processing in Hardware
Salvatore Pontarelli1,2, Roberto Bifulco3, Marco Bonola1,2, Carmelo Cascone4, Marco Spaziani1,2, Valerio Bruschi2, Davide Sanvitto6, Giuseppe Siracusano3, Antonio Capone6, Michio Honda3, Felipe Huici1 and Giuseppe Bianchi1
1Asxbyd, 2CNT, 3NEC Laboratories Europe, 4Open Networking Foundation, 5University of Rome Tor Vergata, 6Politecnico di Milano

NetBricks: Taking the V out of NFV
Aurojit Panda1 Sangjin Han1 Keon Jang1 Melvin Walls1 Sylvia Ratnasamy† Scott Shenker†
1 UC Berkeley 2 Google * ICSI

Abstract
While programming middlebox applications is slow and error-prone, middlebox programming frameworks have emerged to help. However, many frameworks are difficult to use and not well integrated with the rest of the network stack. This paper presents NetBricks, a new approach for building middlebox applications that combines the best features of modern programming frameworks with the performance of custom middleboxes.

ClickNP: Highly Flexible and High Performance Network Processing with Reconfigurable Hardware
Bojie Li††, Kun Tan†, Layong (Larry) Luo†, Yangping Pang†, Rengian Luo††
†† Microsoft Research ‡‡ USTC ¶¶ Microsoft * SJTU

1. INTRODUCTION
Modern multi-tenant datacenters provide shared infrastructure for hosting many different types of services from different customers (e.g., 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: (a) they are not deployable and (b) they are not flexible. 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 can range from tens of microseconds to milliseconds [22, 33, 39]. For many low latency applications (e.g., stock trading), this inflated latency is unacceptable.

Existing software NFs typically use a single-core processor to achieve high performance, which is often the bottleneck. Therefore, software NFs require careful design and optimization to achieve high performance. However, existing software NFs are not flexible, and it is difficult to change their performance characteristics in software.

ClickNP is a novel approach for building flexible and high performance network processing systems. It is based on a hardware platform that can be reconfigured at runtime to support different applications. This approach allows for high performance and flexibility, which is essential for modern network infrastructure.

ClickNP consists of two main components: (a) a hardware platform that can be reconfigured at runtime, and (b) a software framework that can be used to build middlebox applications. The hardware platform is based on a reconfigurable system on chip (SoC) that can be reconfigured at runtime to support different applications. The software framework is based on a high-level abstraction that can be used to build middlebox applications.

The ClickNP framework provides a high-level abstraction for building middlebox applications. It allows developers to build middlebox applications that are scalable and flexible. The ClickNP framework is based on a high-level abstraction that can be used to build middlebox applications. It allows developers to build middlebox applications that are scalable and flexible.

The ClickNP framework is based on a high-level abstraction that can be used to build middlebox applications. It allows developers to build middlebox applications that are scalable and flexible.

The ClickNP framework is based on a high-level abstraction that can be used to build middlebox applications. It allows developers to build middlebox applications that are scalable and flexible.

The ClickNP framework is based on a high-level abstraction that can be used to build middlebox applications. It allows developers to build middlebox applications that are scalable and flexible.
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 $>10$MPPS

• **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.
  – *Achieved by scheduling policies (left largely to future work)*
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.
  – Achieved by scheduling policies (left largely to future work)
Current Solution

- vSwitch
- VM/Container
- VM/Container
- VM/Container
- NIC
- NIC

Memory Isolation
Packet Isolation
Performance
Current Solution

- vSwitch
- NIC
- VM/Container
- VM/Container
- VM/Container

Memory Isolation
Packet Isolation
Performance
Current Solution

- vSwitch
- NIC ...
- NIC

- VM/Container
- VM/Container
- VM/Container

- Memory Isolation
- Packet Isolation
- Performance
Current Solution

- vSwitch
- NIC
- VM/Container
- NIC
- VM/Container
- VM/Container

✔ Memory Isolation
Packet Isolation
Performance
Current Solution

- vSwitch
- VM/Container
- VM/Container
- VM/Container
- NIC
- NIC

- Memory Isolation
- Packet Isolation
- Performance
Current Solution

- vSwitch
- VM/Container
- VM/Container
- VM/Container

NIC...NIC

- Memory Isolation
- Packet Isolation
- Performance
Current Solution

- vSwitch
- Copy
- VM/Container
- VM/Container
- VM/Container

NIC...

- Memory Isolation
- Packet Isolation
- Performance
Current Solution

- vSwitch
- Copy
- VM/Container
- VM/Container
- VM/Container

- NIC
- NIC

✓ Memory Isolation

Packet Isolation

Performance
Current Solution

- vSwitch
- Copy
- NIC
- VM/Container
- VM/Container
- VM/Container

- Memory Isolation
- Packet Isolation
- Performance
Current Solution

- Memory Isolation
- Packet Isolation

- Performance
Isolation costs Performance

No Isolation

Processing Rate (Mpps)
Isolation costs Performance

- No Isolation
- OVS VM

Processing Rate (Mpps)
Isolation costs Performance

- No Isolation
- OVS VM
- BESS VM

Processing Rate (Mpps)
Isolation costs Performance

![Bar Chart]

- No Isolation
- OVS VM
- BESS VM
- BESS Container

Processing Rate (Mpps)
NetBricks Runtime Architecture

Single Process Space

ZCSI Scheduler

DPDK Poll for I/O

NICs
NetBricks Runtime Architecture

Single Process Space

ZCSI Scheduler

Function Call

DPDK Poll for I/O

NICs
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

<table>
<thead>
<tr>
<th>Packet Processing</th>
<th>Control Flow</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse/Deparse</td>
<td>Group By</td>
<td>Bounded Consistency</td>
</tr>
<tr>
<td>Transform</td>
<td>Shuffle</td>
<td></td>
</tr>
<tr>
<td>Filter</td>
<td>Merge</td>
<td></td>
</tr>
</tbody>
</table>

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 thoughts?

• What did you like about the paper?

• What are its limitations?
Evolution of middleboxes

Dedicated hardware

Packets

ASIC

Middleboxes

Need for flexibility

Software

Packets

CPU

Network functions

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

Dedicated hardware

Packets
ASIC

Middleboxes

Need for flexibility

Software

Packets
CPU

Network functions

Need for performance

Programmable HW

Packets
e.g. FPGA

ClickNP, SIGCOMM’16
FlowBlaze, NSDI’19
TEA, SIGCOMM’20
Evolution of middleboxes

Dedicated hardware

Packets
ASIC

Middleboxes

Need for flexibility

Software

Packets
CPU

Network functions

Programmable HW

Packets
e.g. FPGA

Need for performance

ClickNP, SIGCOMM’16
FlowBlaze, NSDI’19
TEA, SIGCOMM’20
Upcoming classes

• No class on Nov 16th.
  – Optional reading on network edge.
  – 5G systems microbook is a useful resource (linked on the webpage).

• Nov 18 and Nov 30: your presentations (each up to 6mins long).

• Fiday, Dec 2\textsuperscript{nd}:
  – Wrapping up
  – Quiz for \textit{bonus} class participation scores.

• Monday, Dec 5\textsuperscript{th}: Final projects due (will update on course website).

• Dec 7\textsuperscript{th}: Final project presentation. Details TBA.