LemonNFV: Consolidating Heterogeneous Network Functions at Line Speed

Hao Li[1], **Yihan Dang[1]**, Guangda Sun[1,2], Guyue Liu[3],
Danfeng Shan[1], Peng Zhang[1]

西安交通大學
XI'AN JIAOTONG UNIVERSITY

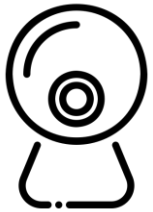
NUS
National University
of Singapore

NYU 上海

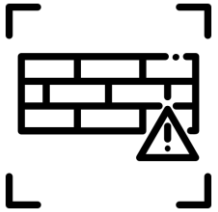
SHANGHAI 纽约大学

Virtualized Network Function (VNF)

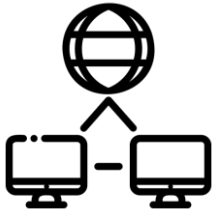
DPI



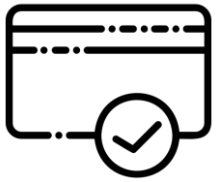
IDS



Firewall



NAT



ACL

VPN

Load Balancer

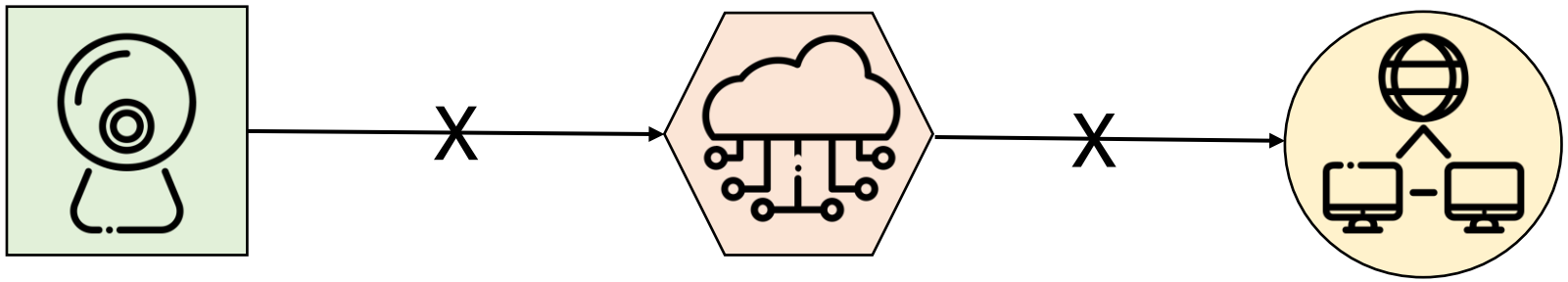
Virtualized Network Function (VNF)

IDS

Load Balancer

NAT

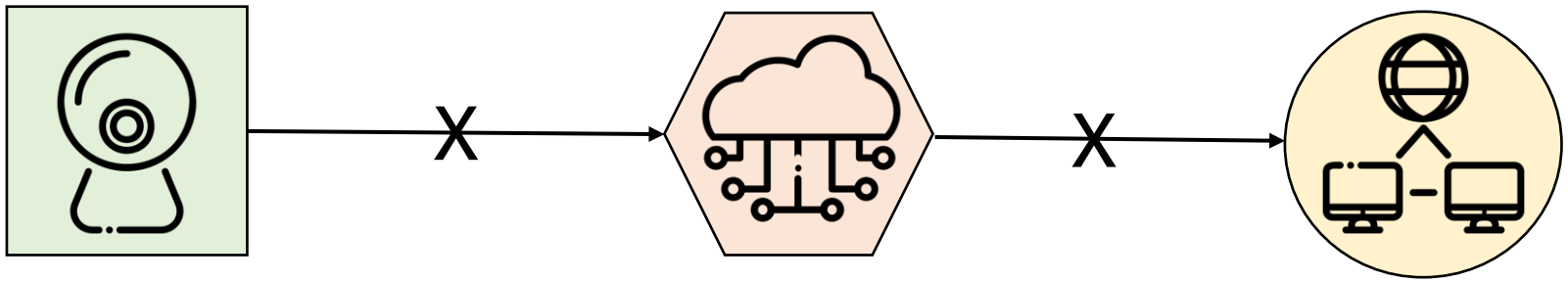
Heterogeneous NFs Are Not Interoperable



Programming language *Execution model* *State & Packet Abstraction*

Different shapes signify different vendors, and thus interfaces, abstractions, etc.

Heterogeneous NFs Are Not Interoperable



Snort
[LISA ‘99]

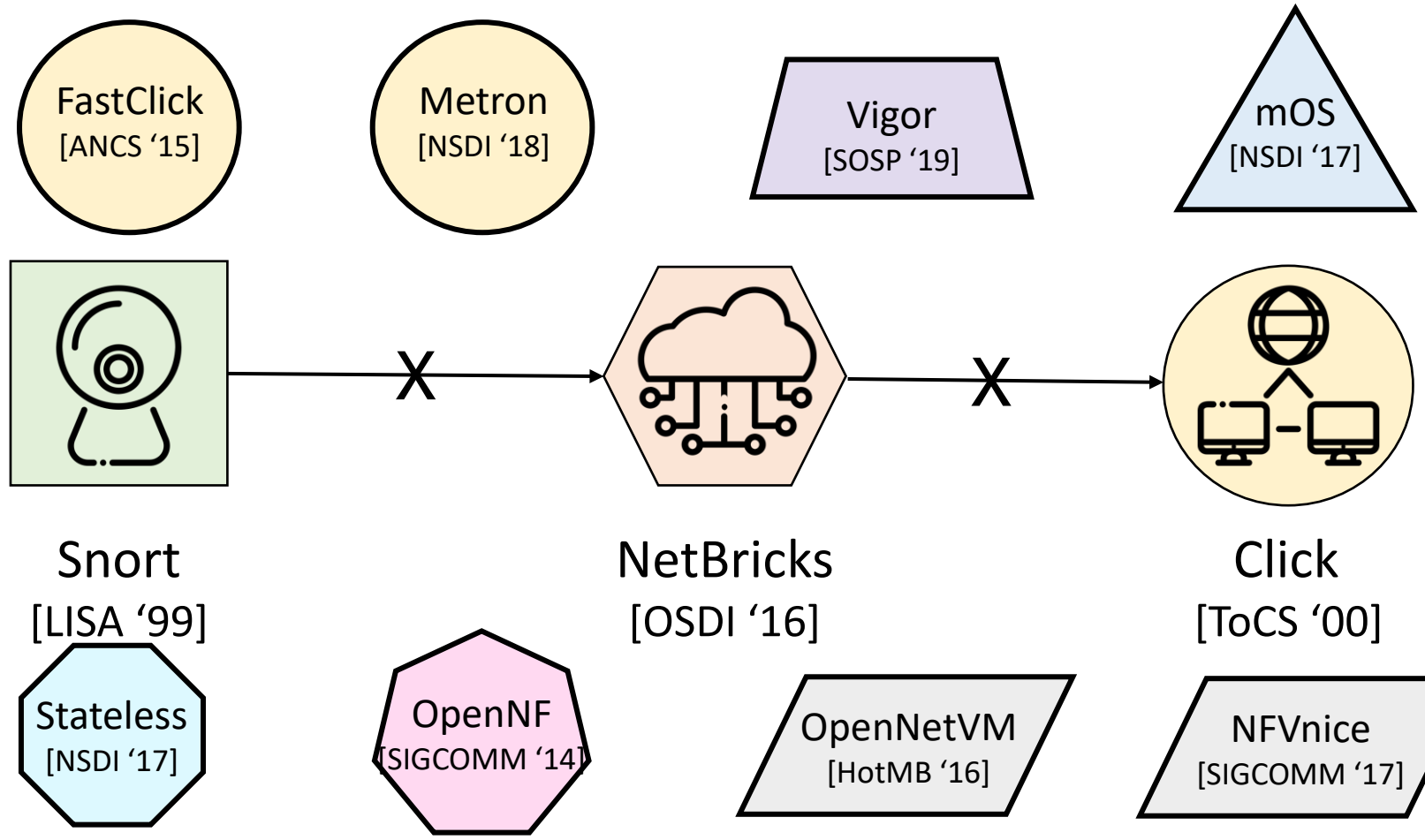
NetBricks
[OSDI ‘16]

Click
[ToCS ‘00]

Programming language *Execution model* *State & Packet Abstraction*

Different shapes signify different vendors, and thus interfaces, abstractions, etc.

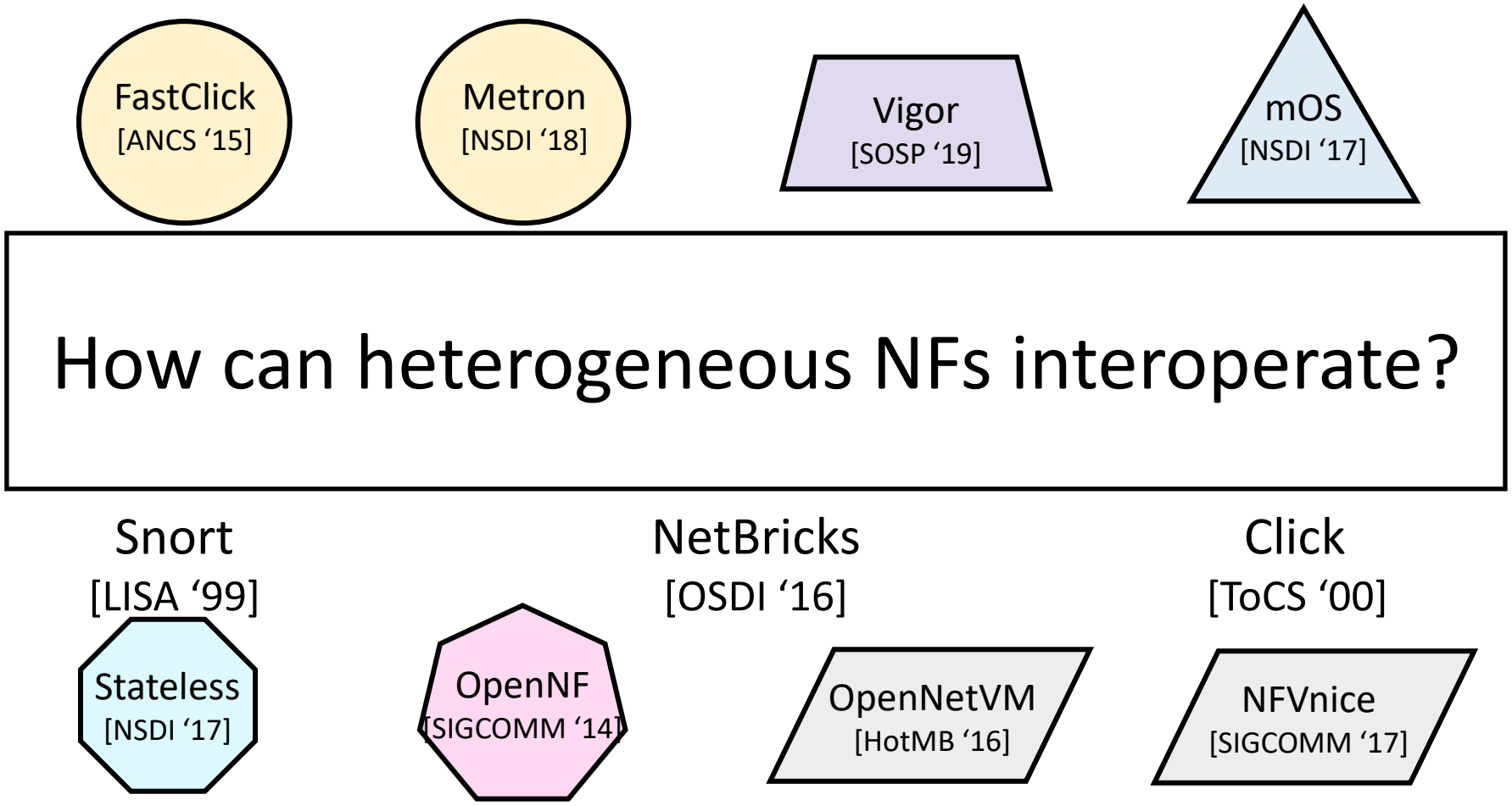
Heterogeneous NFs Are Not Interoperable



Programming language *Execution model* *State & Packet Abstraction*

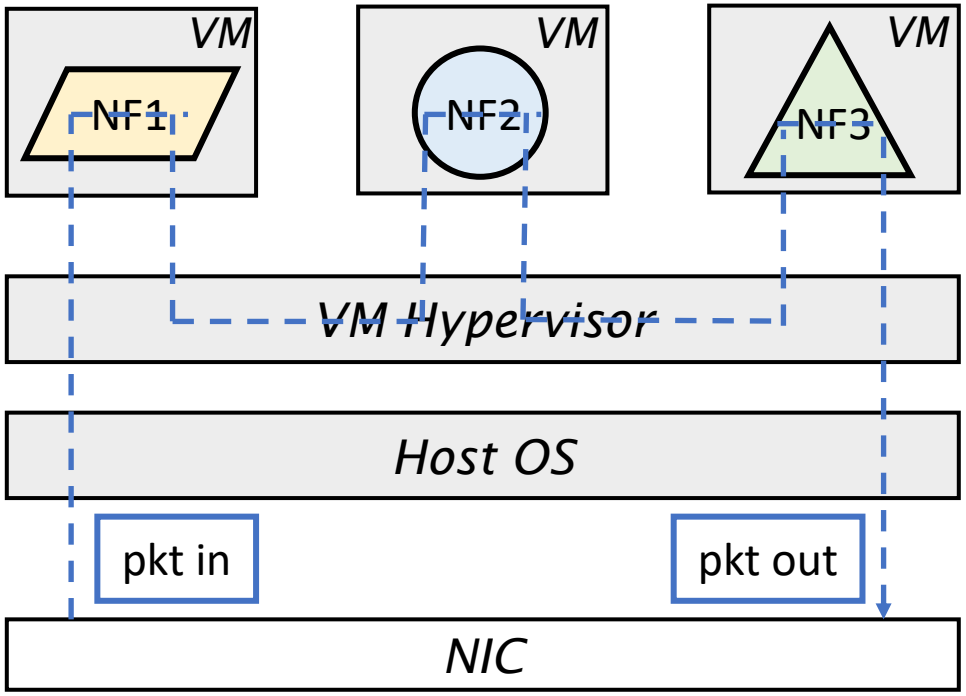
Different shapes signify different vendors, and thus interfaces, abstractions, etc.

Heterogeneous NFs Are Not Interoperable



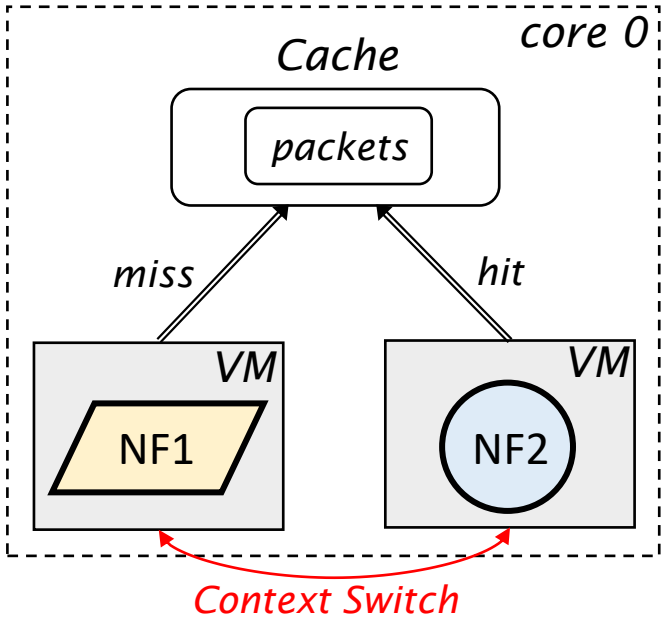
Different shapes signify different vendors, and thus interfaces, abstractions, etc.

Solution 1: Virtualization



Dilemma: Context Switch vs Inter-Core Traffic

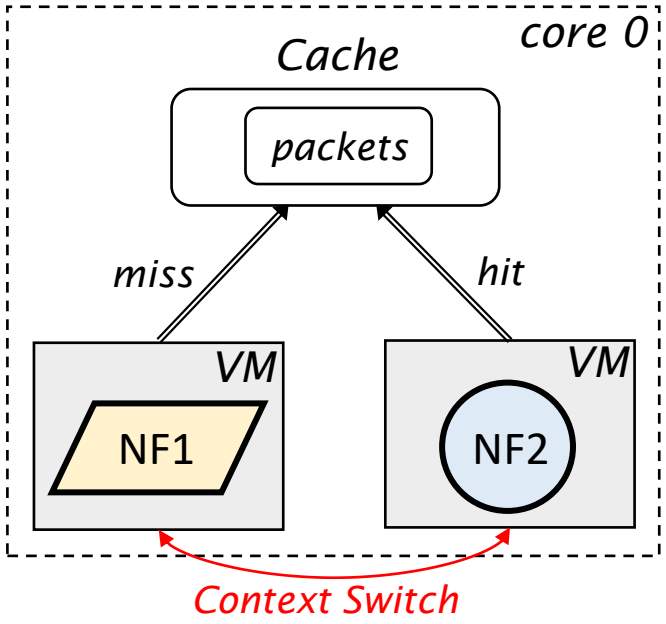
Dilemma: Context Switch vs Inter-Core Traffic

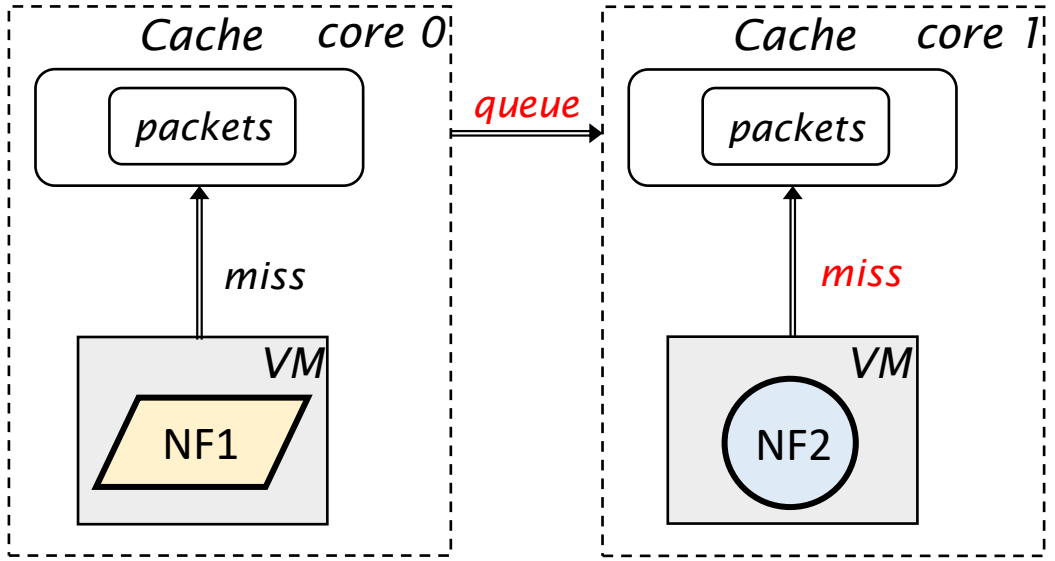


- Scheduling instances on the same core
- Related Work: Quadrant *[SoCC '22]*
  - Reportedly 41.4% more latency[1]

[1] J. Wang, T. Lévai, Z. Li, M. A. M. Vieira, R. Govindan, and B. Raghavan. Quadrant: a cloud-deployable NF virtualization platform. In SoCC '22

Dilemma: Context Switch vs Inter-Core Traffic



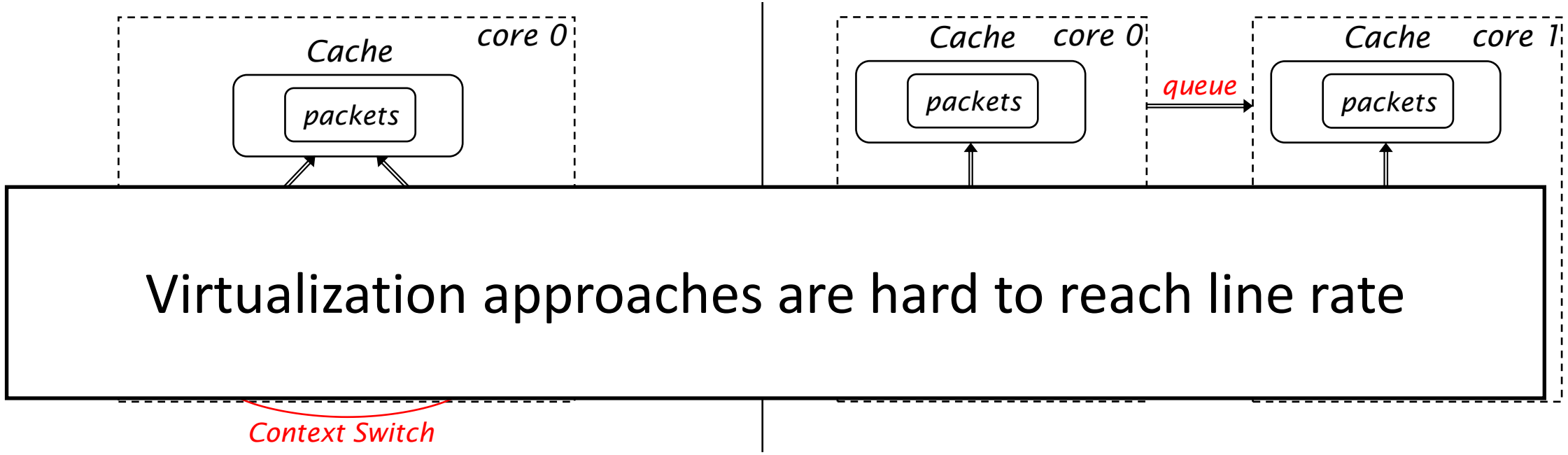


- Scheduling instances on the same core
- Related Work: Quadrant *[SoCC '22]*
  - Reportedly 41.4% more latency[1]

- Pinning instances on dedicated cores
- Related Work: OpenNetVM *[HotMB '16]*
  - Reportedly at least 121.2% more latency[2]

[1] J. Wang, T. Lévai, Z. Li, M. A. M. Vieira, R. Govindan, and B. Raghavan. Quadrant: a cloud-deployable NF virtualization platform. In SoCC '22
[2] C. Sun, J. Bi, Z. Zheng, H. Yu, and H. Hu. NFP: Enabling Network Function Parallelism in NFV. In SIGCOMM '17

Dilemma: Context Switch vs Inter-Core Traffic



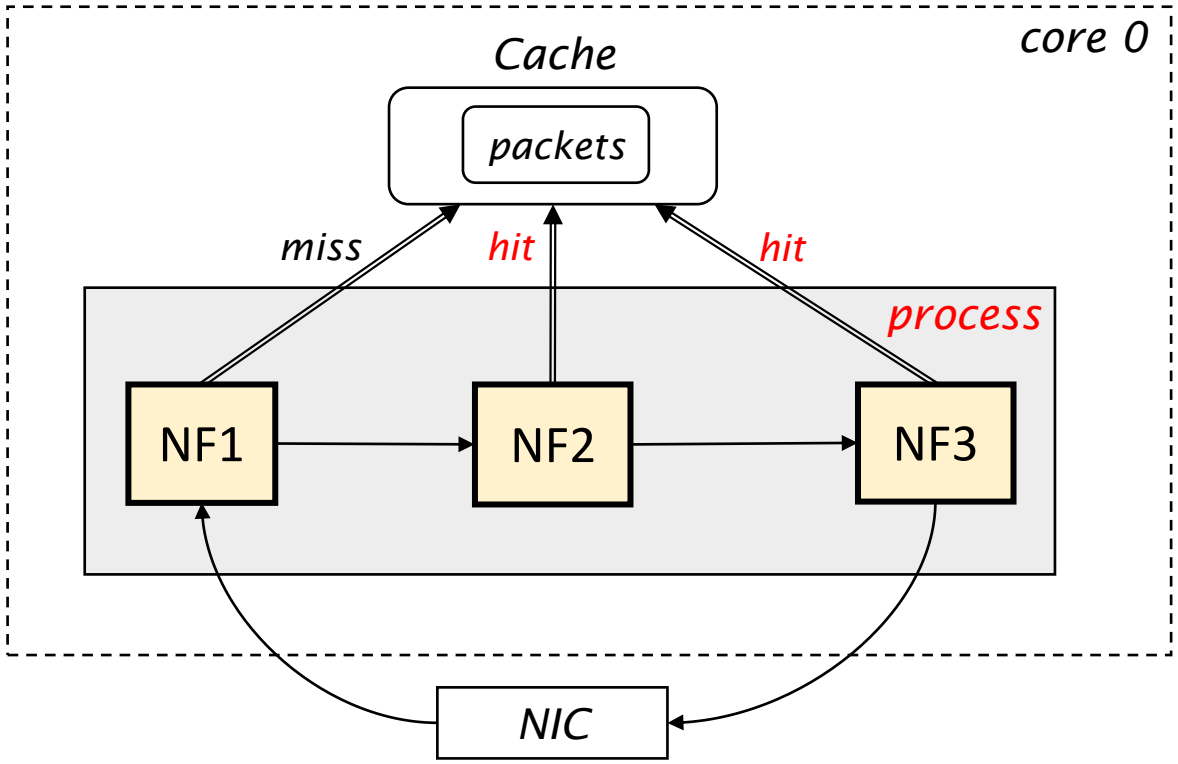
- Scheduling instances on the same core
- Related Work: Quadrant *[SoCC '22]*
  - Reportedly 41.4% more latency[1]

- Pinning instances on dedicated cores
- Related Work: OpenNetVM *[HotMB '16]*
  - Reportedly at least 121.2% more latency[2]

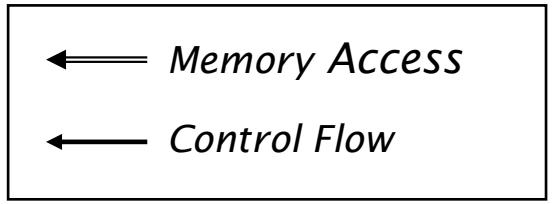
[1] J. Wang, T. Lévai, Z. Li, M. A. M. Vieira, R. Govindan, and B. Raghavan. Quadrant: a cloud-deployable NF virtualization platform. In SoCC '22
[2] C. Sun, J. Bi, Z. Zheng, H. Yu, and H. Hu. NFP: Enabling Network Function Parallelism in NFV. In SIGCOMM '17

Solution 2: Consolidation

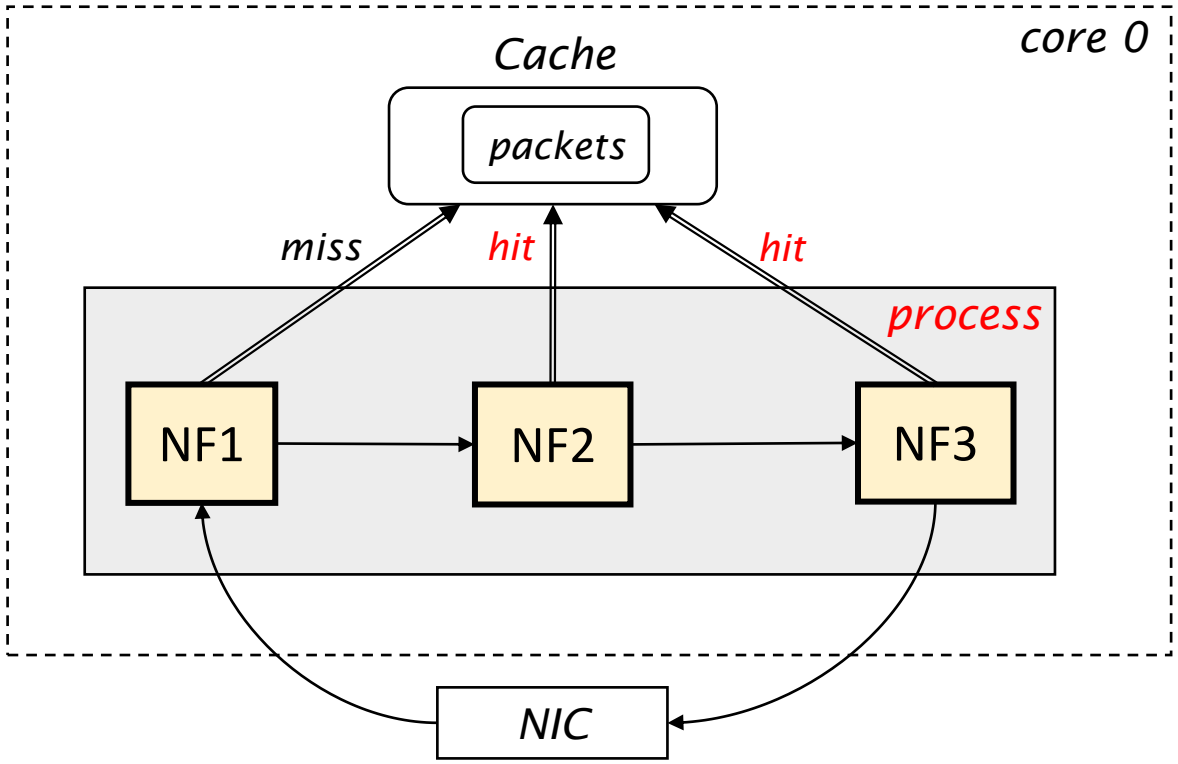
Solution 2: Consolidation



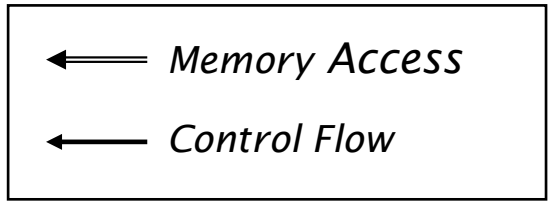
- Fusing all NFs into one process
- No context switching or inter-core traffic



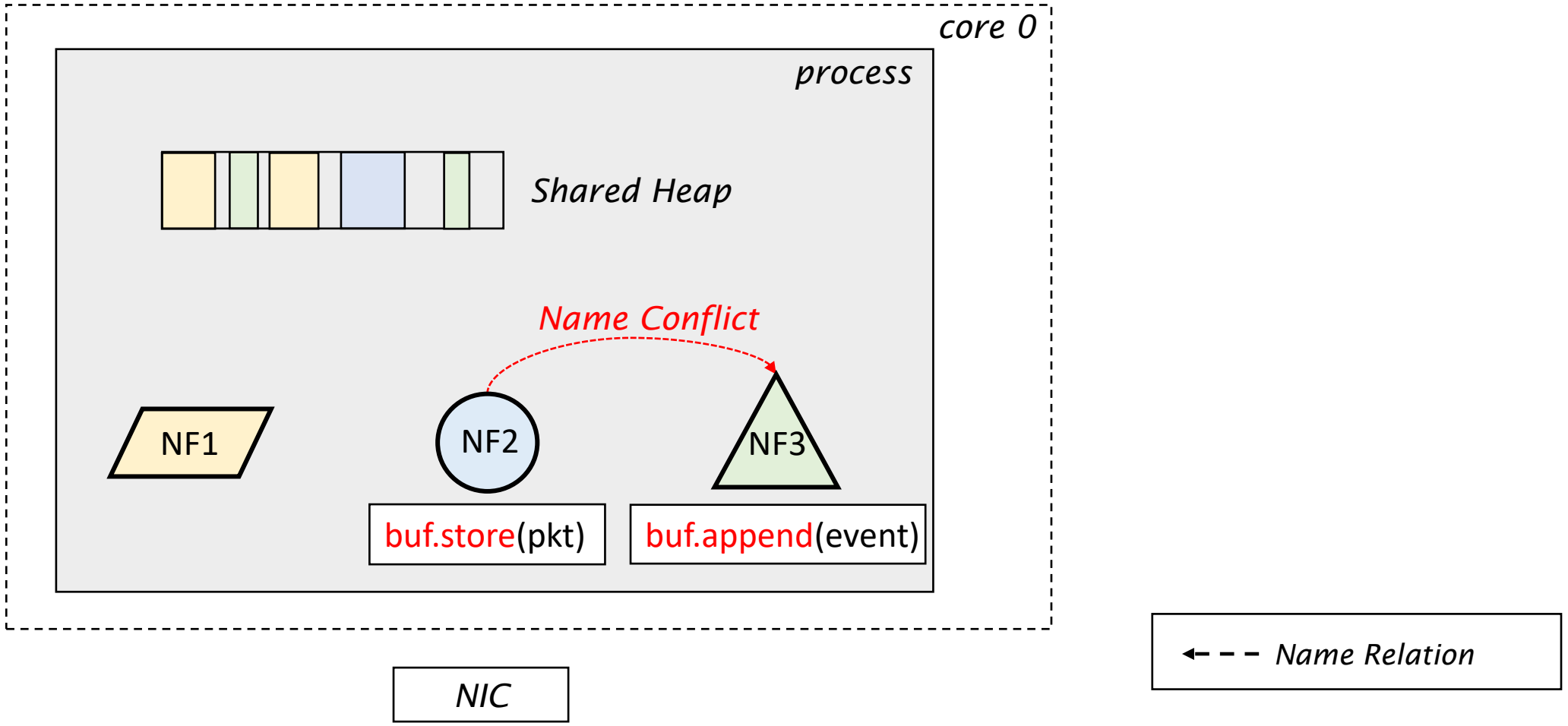
Solution 2: Consolidation



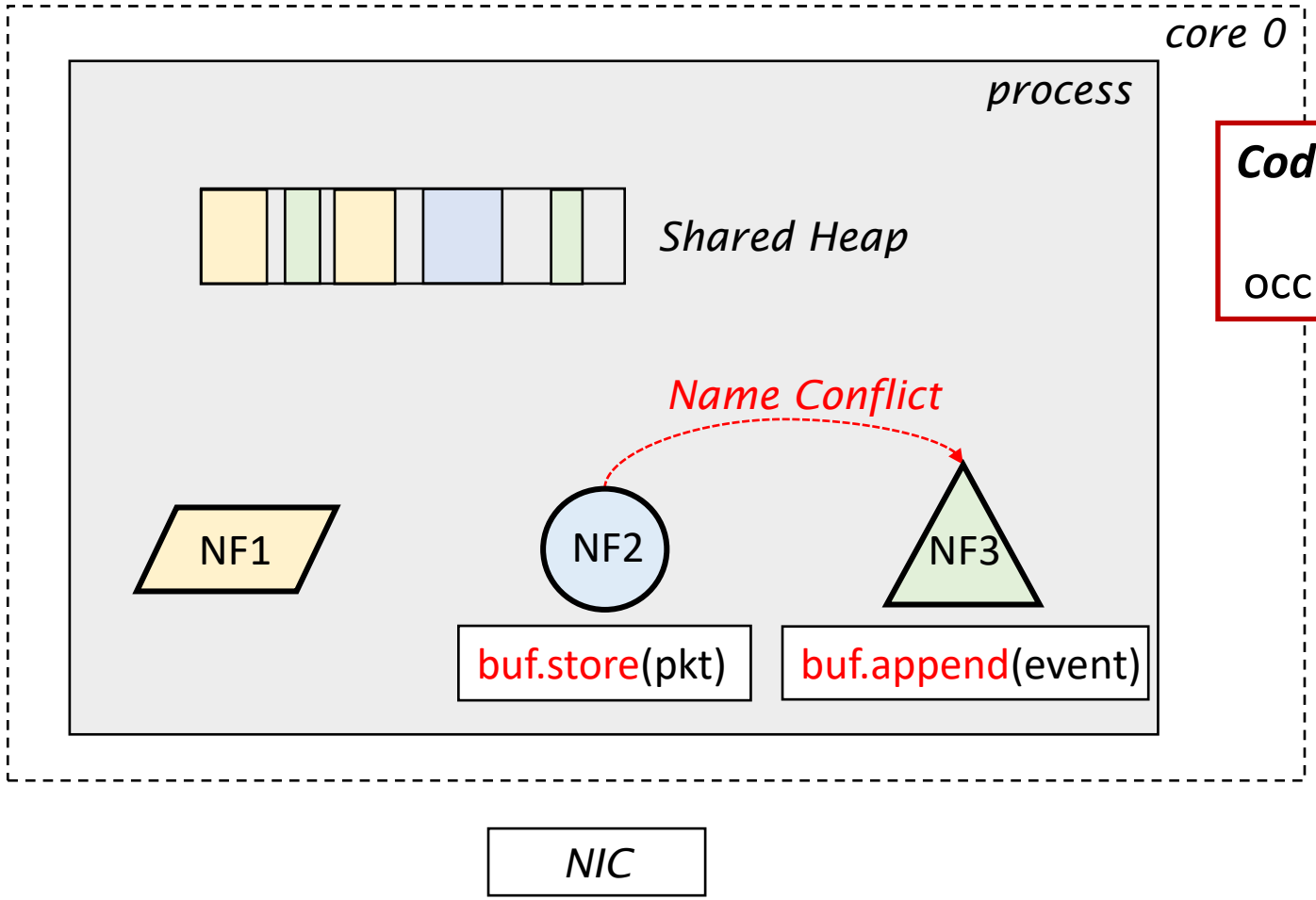
- Fusing all NFs into one process
- No context switching or inter-core traffic
- Requiring huge code modification on heterogeneous NFs



Problem #1: Namespace Conflict



Problem #1: Namespace Conflict

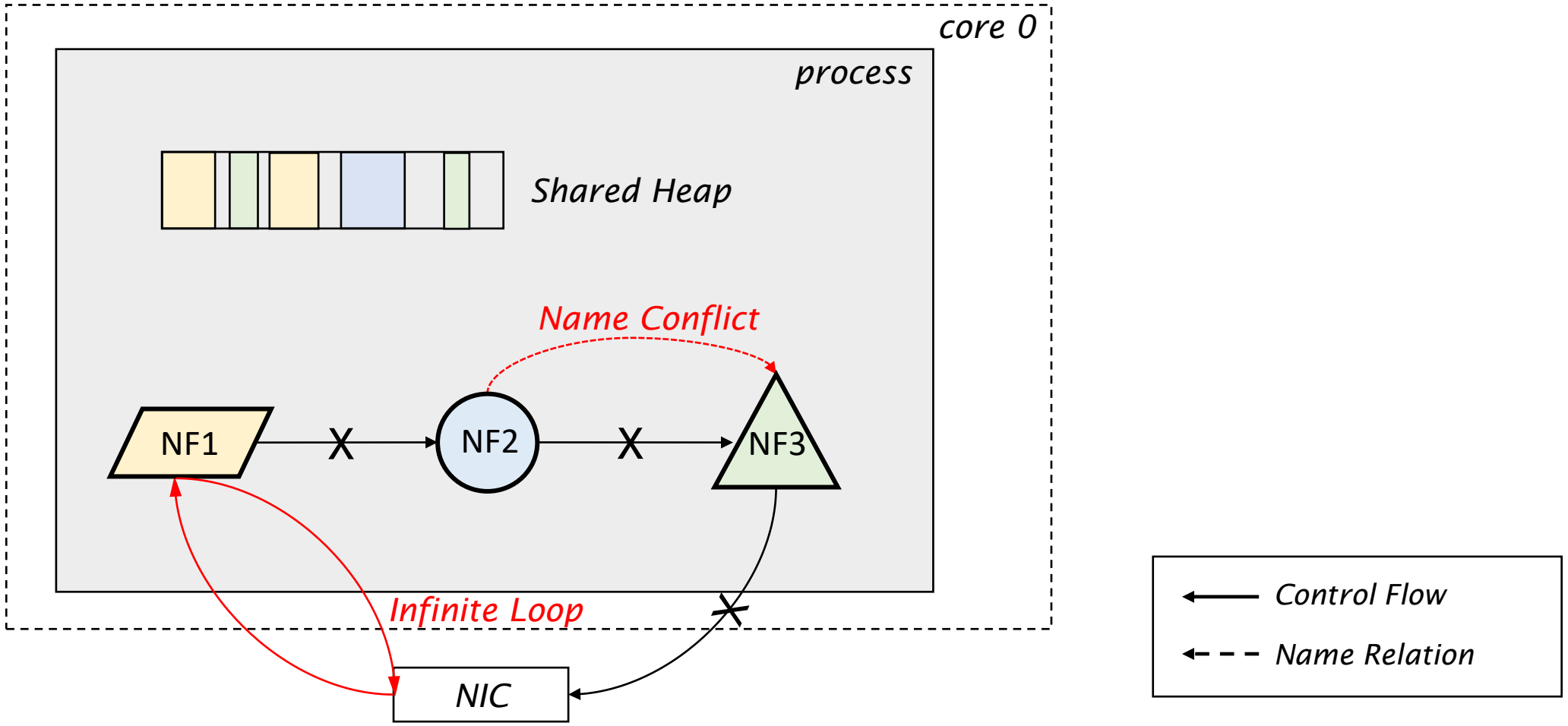


Code Modification

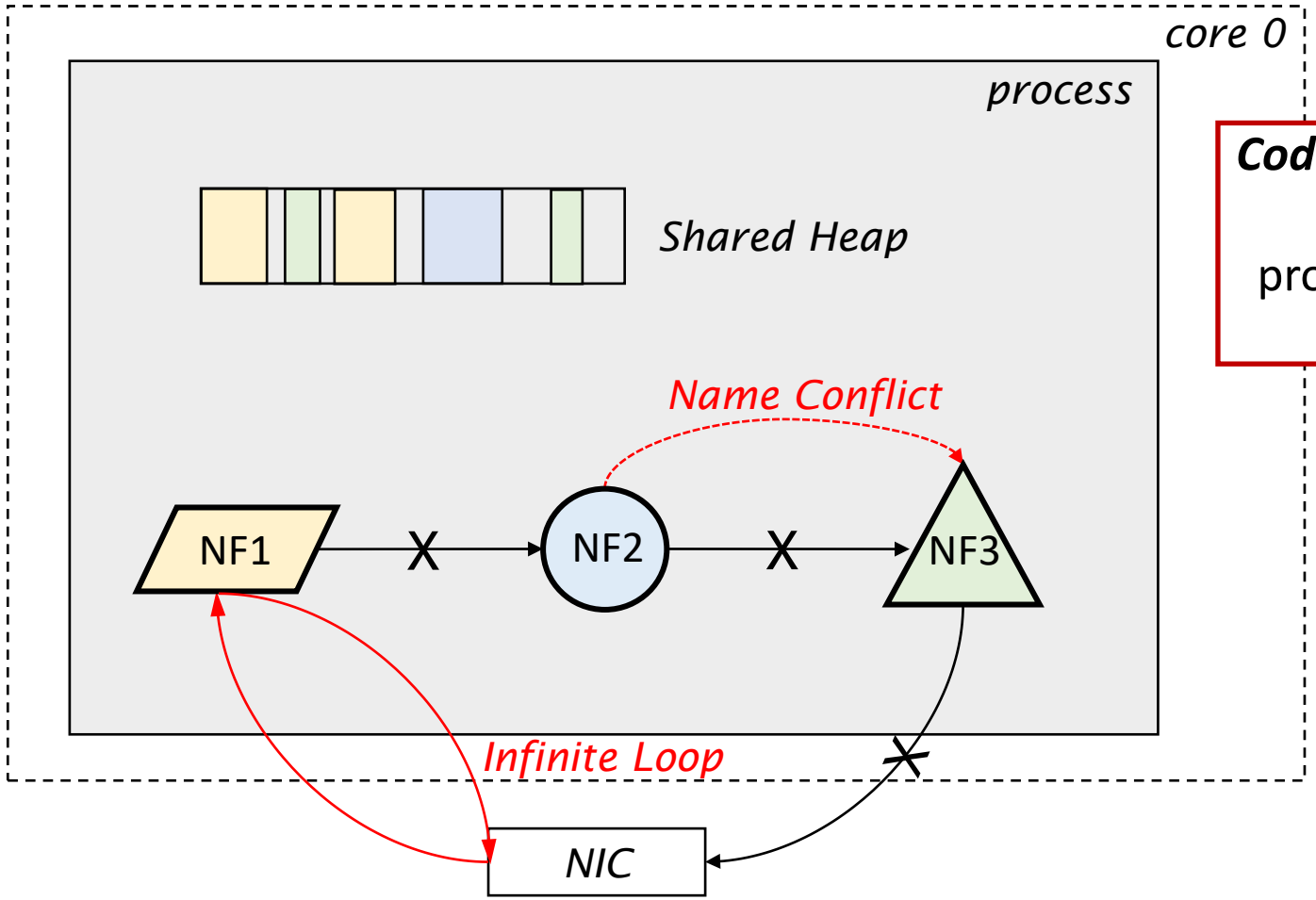
check and replace every occurrence of conflicting names

Name Relation

Problem #2: Private Control Flow

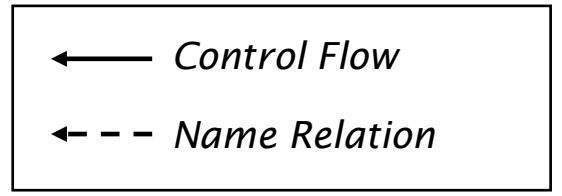


Problem #2: Private Control Flow

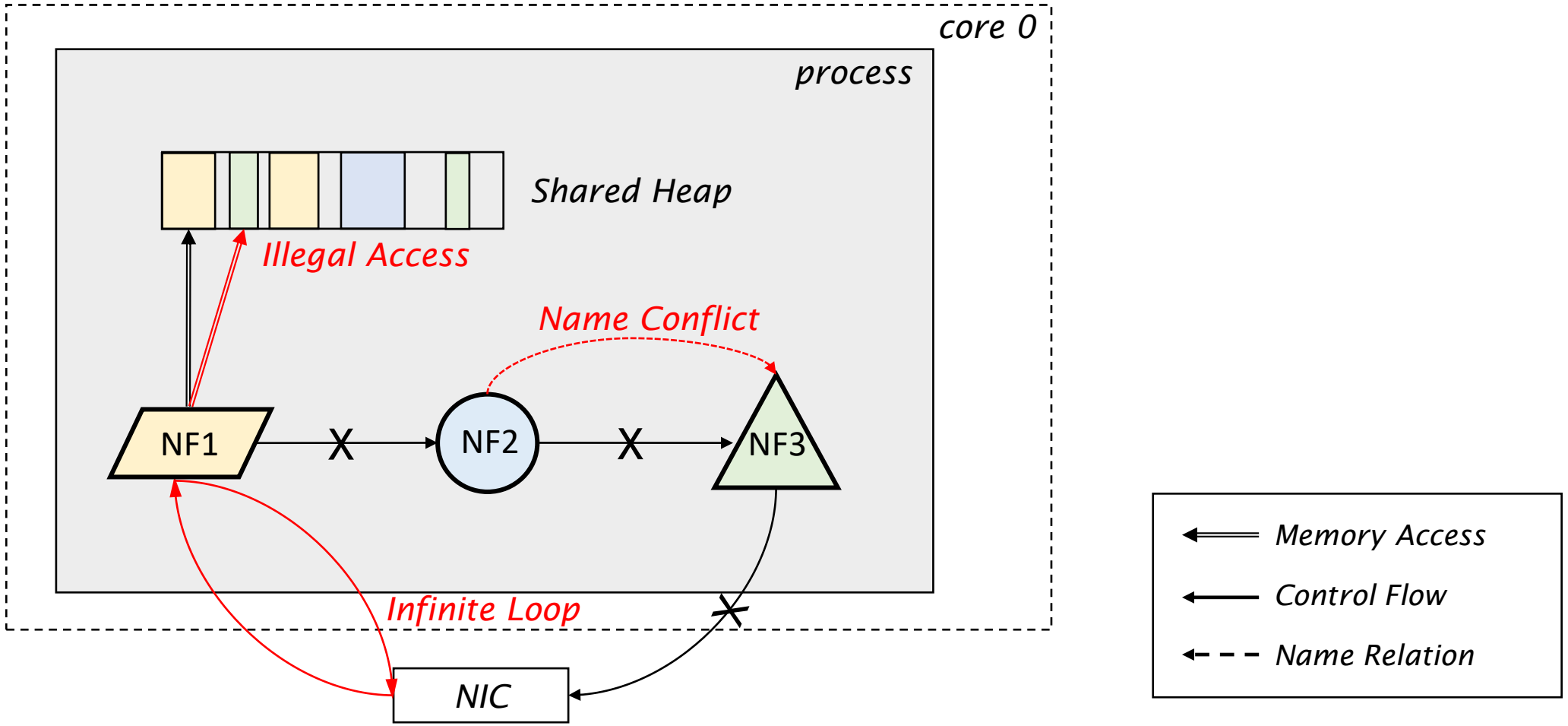


Code Modification

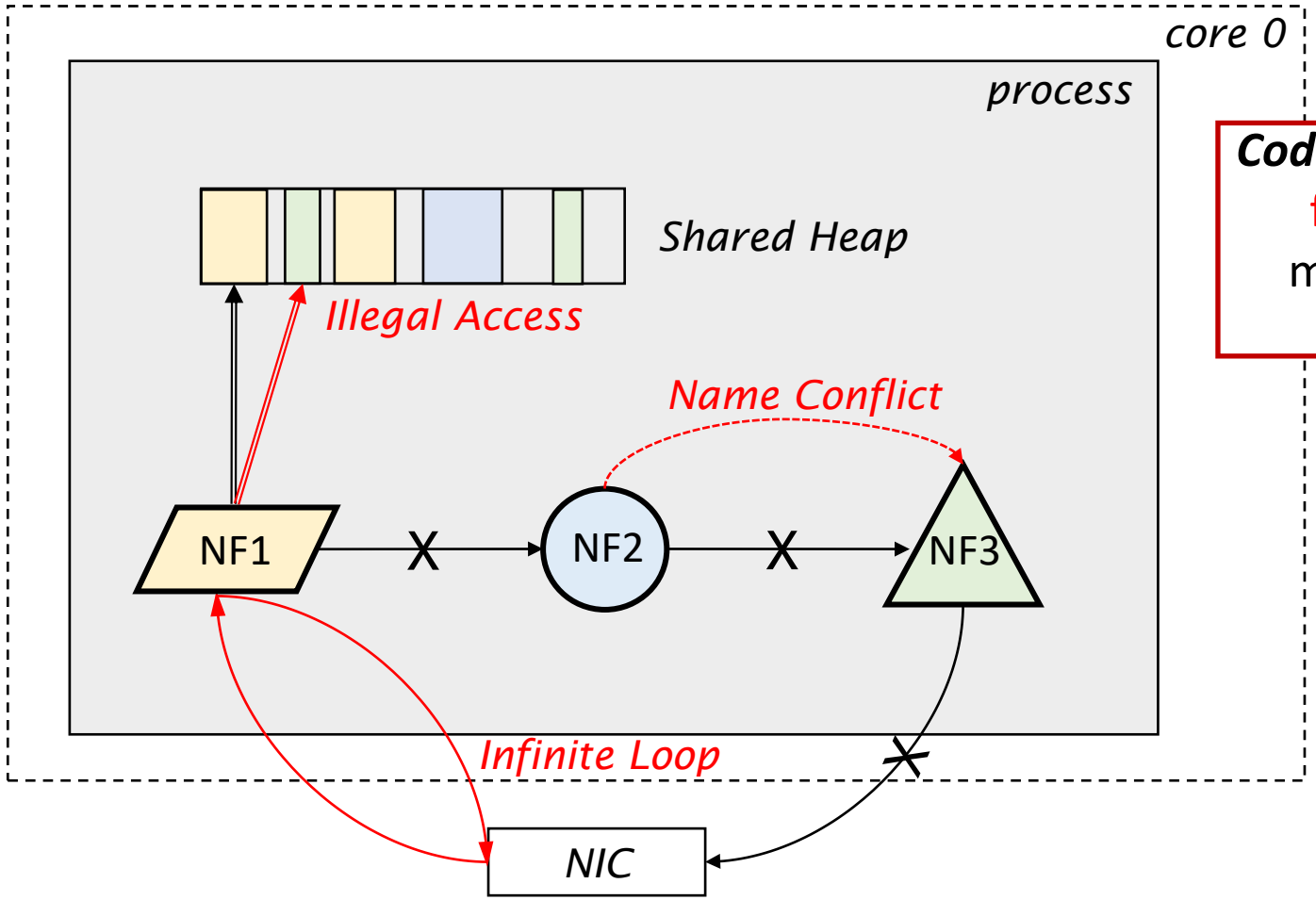
find and extract packet processing logic, and rewrite it without loop



Problem #3: Illegal Memory Access

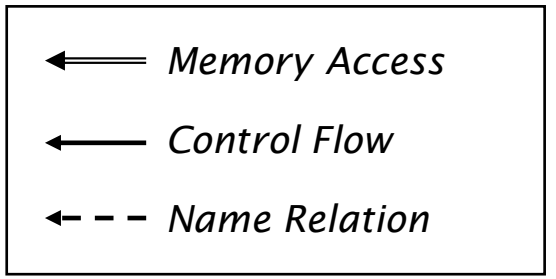


Problem #3: Illegal Memory Access

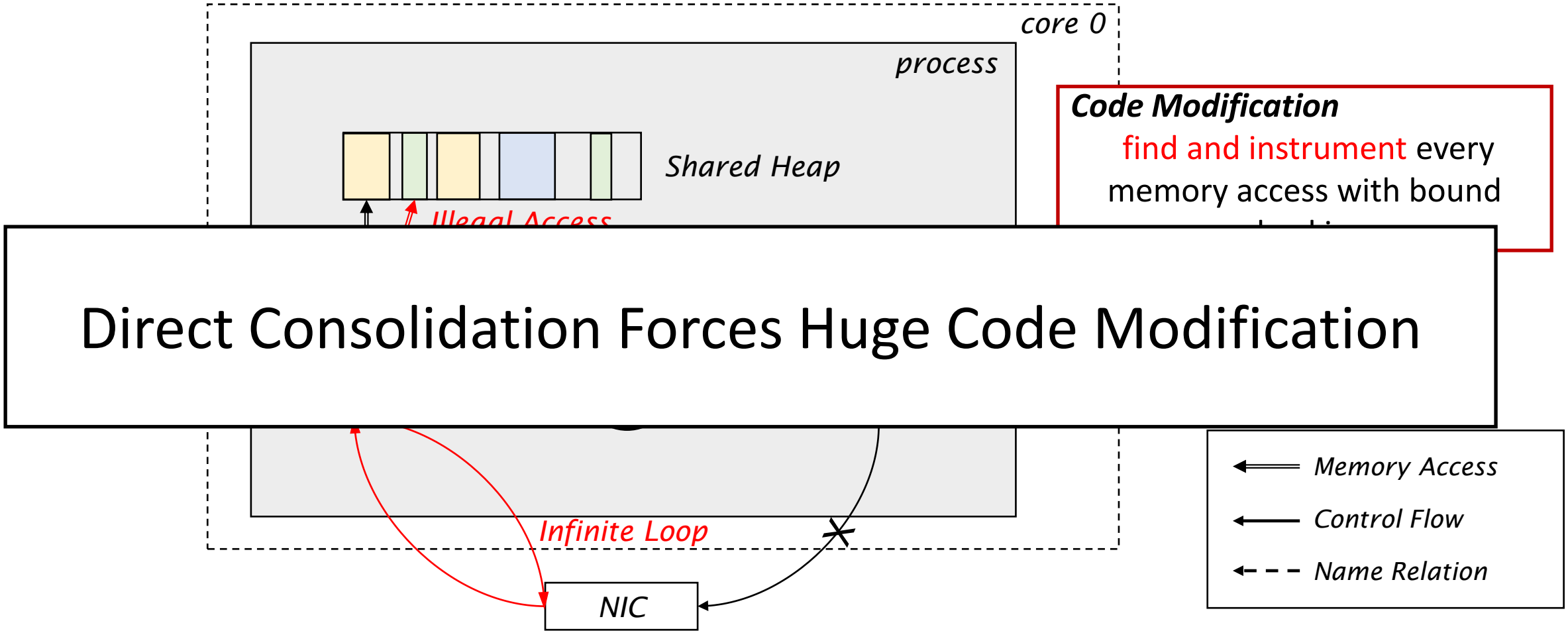


Code Modification

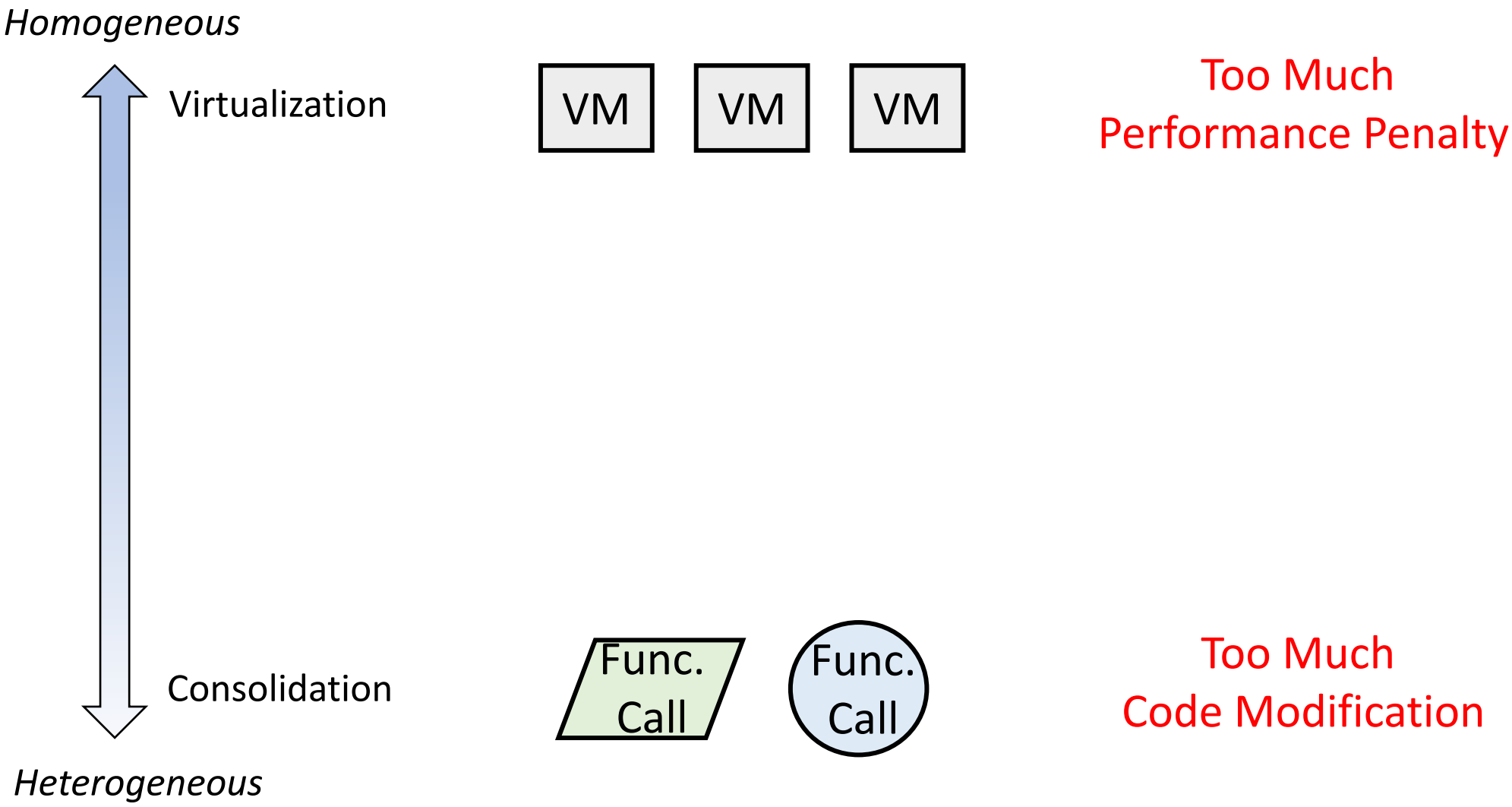
find and instrument every memory access with bound checking



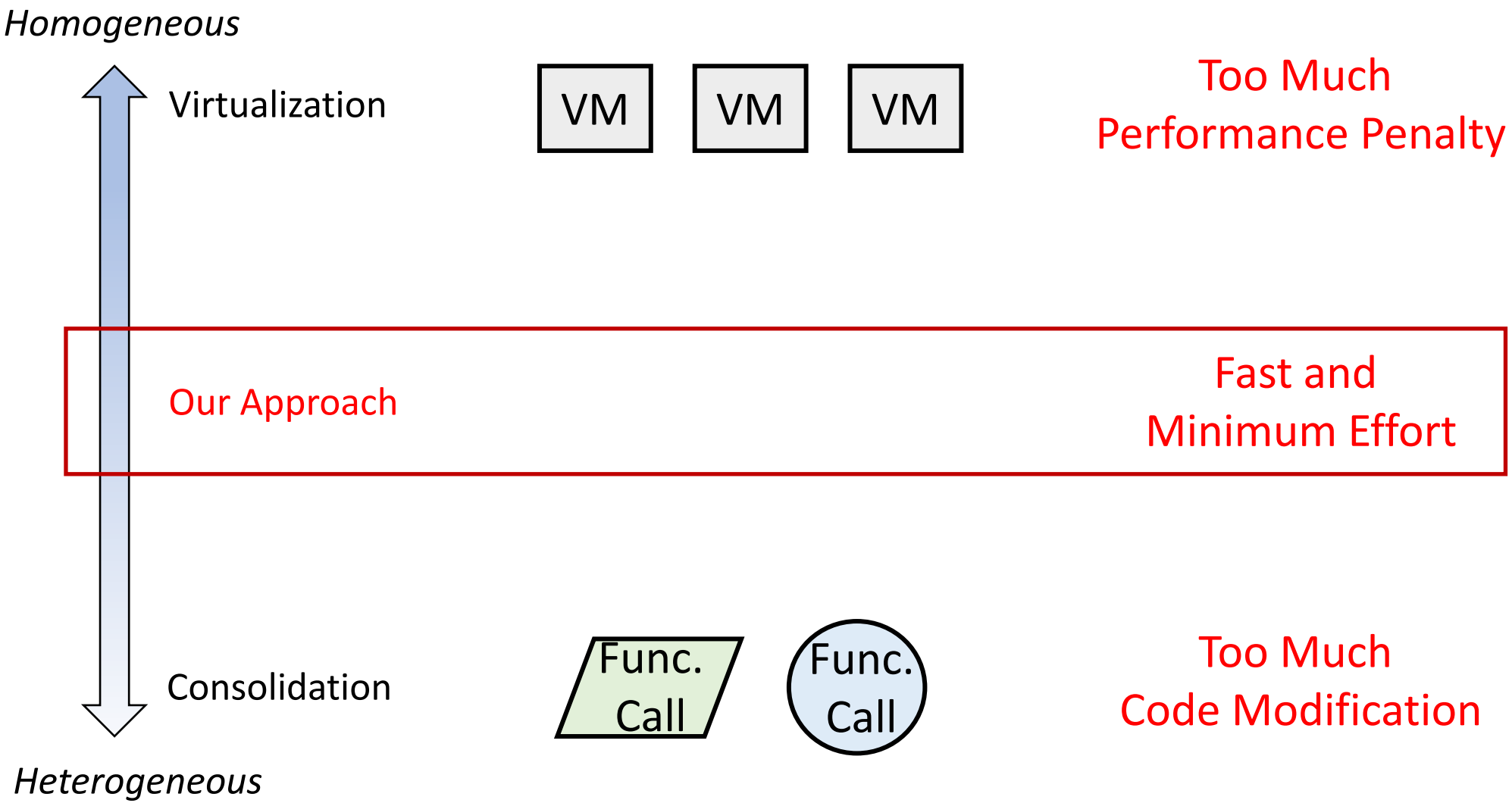
Problem #3: Illegal Memory Access



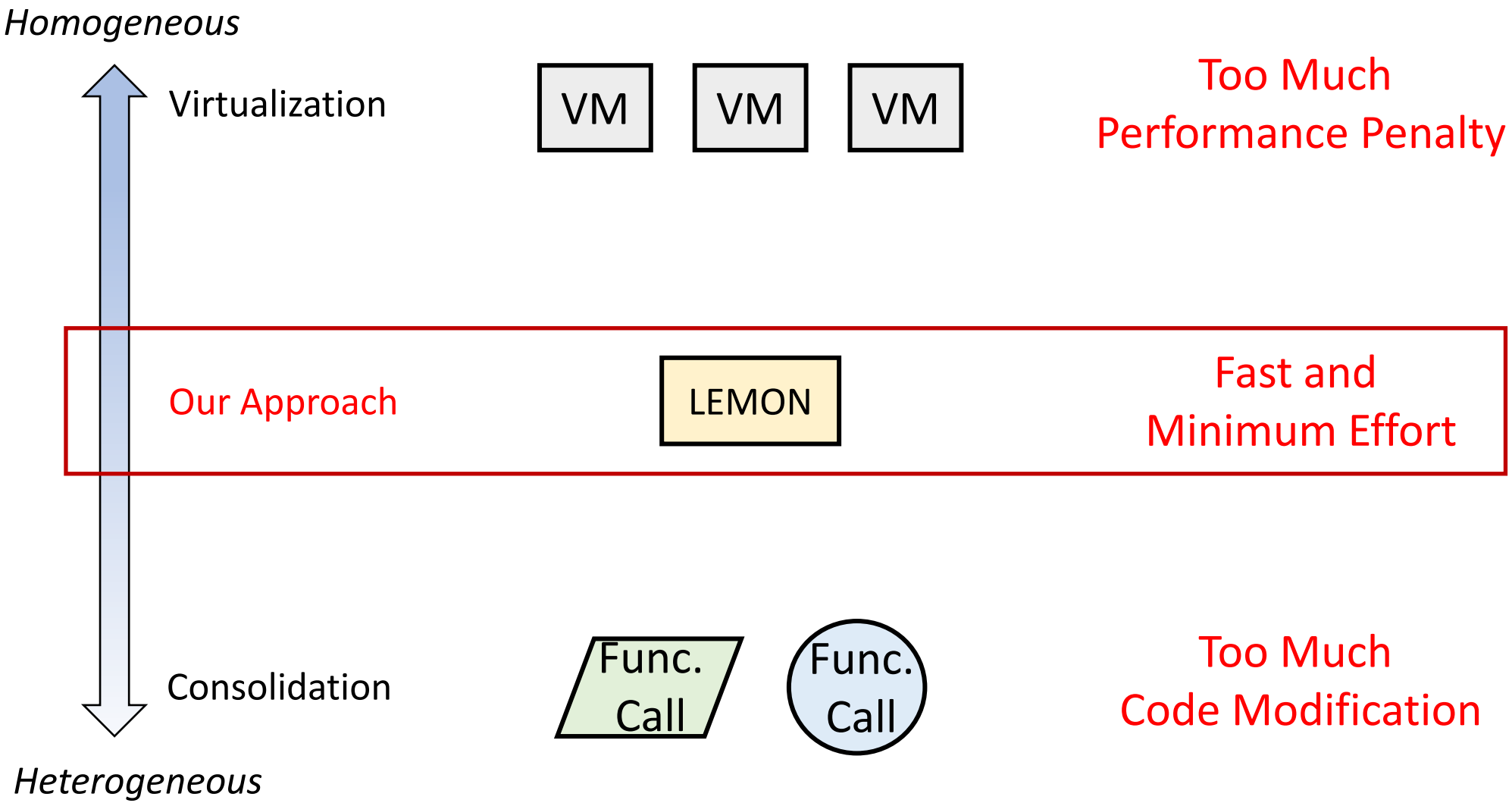
Takeaway on Two Approaches



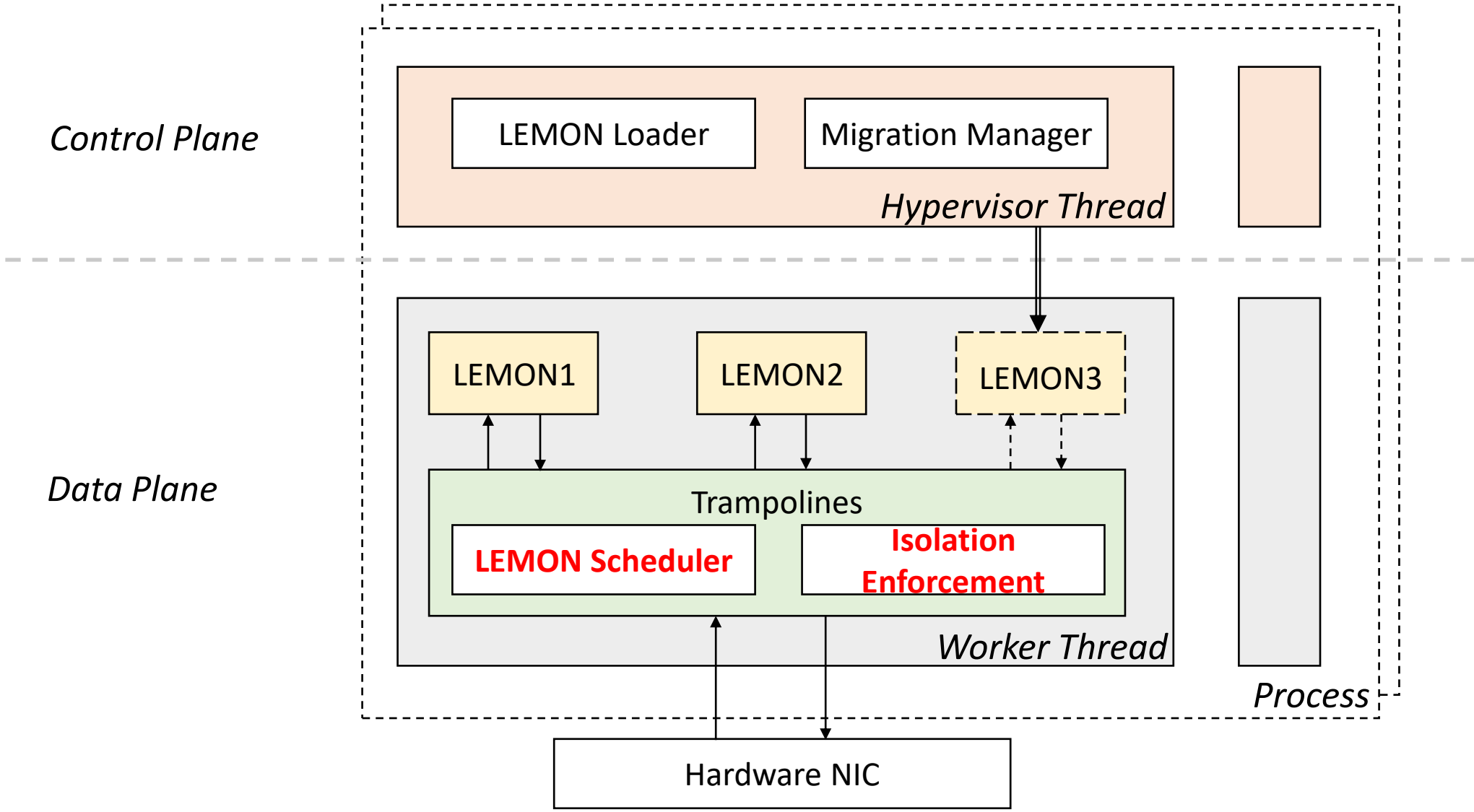
Our Insight


Homogeneous
Virtualization
VM
VM
VM
Too Much
Performance Penalty
Our Approach
Fast and
Minimum Effort
Consolidation
Func.
Call
Func.
Call
Too Much
Code Modification
Heterogeneous

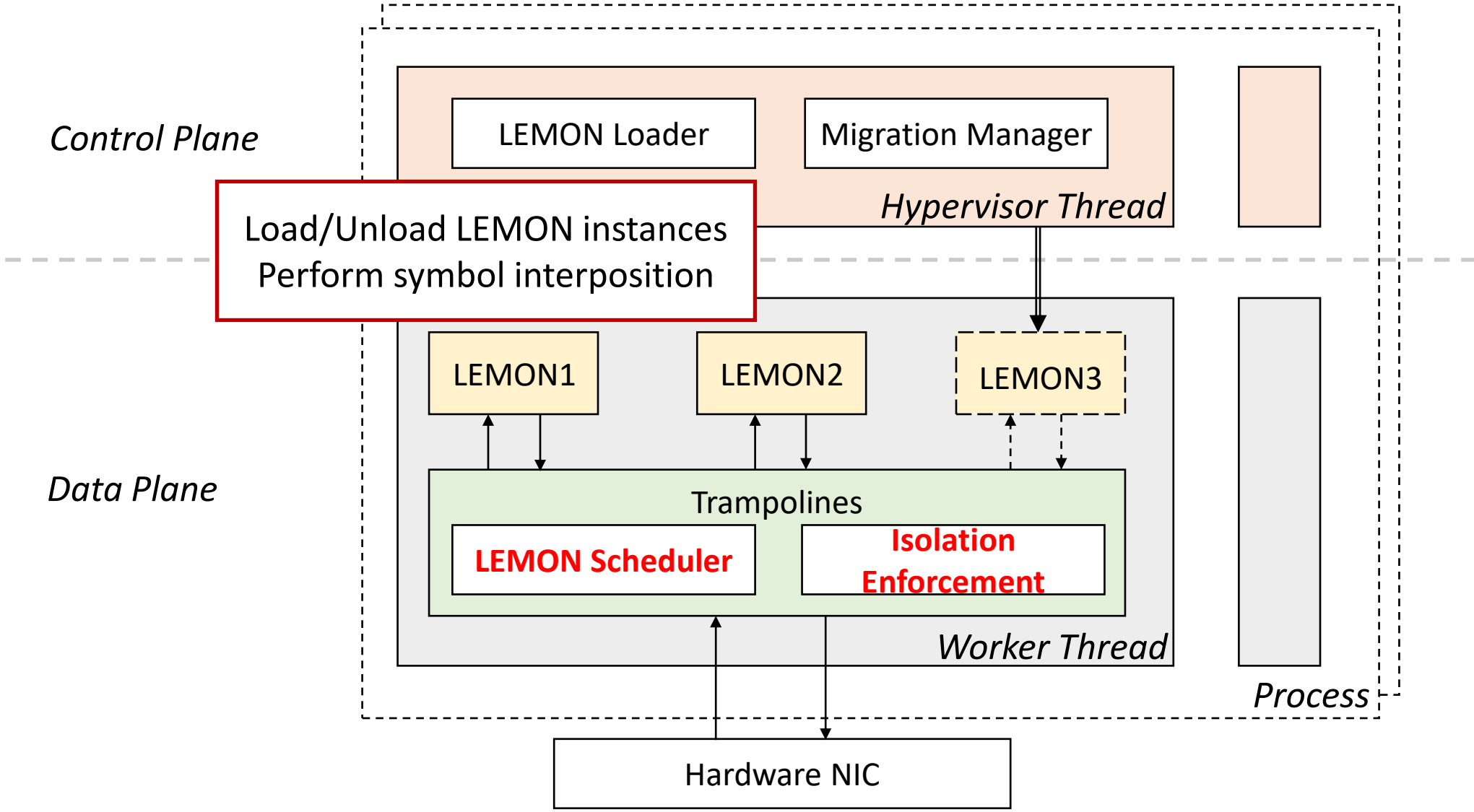
Our Insight


Homogeneous
Virtualization
VM
VM
VM
Too Much
Performance Penalty
Our Approach
LEMON
Fast and
Minimum Effort
Consolidation
Func.
Call
Func.
Call
Too Much
Code Modification
Heterogeneous

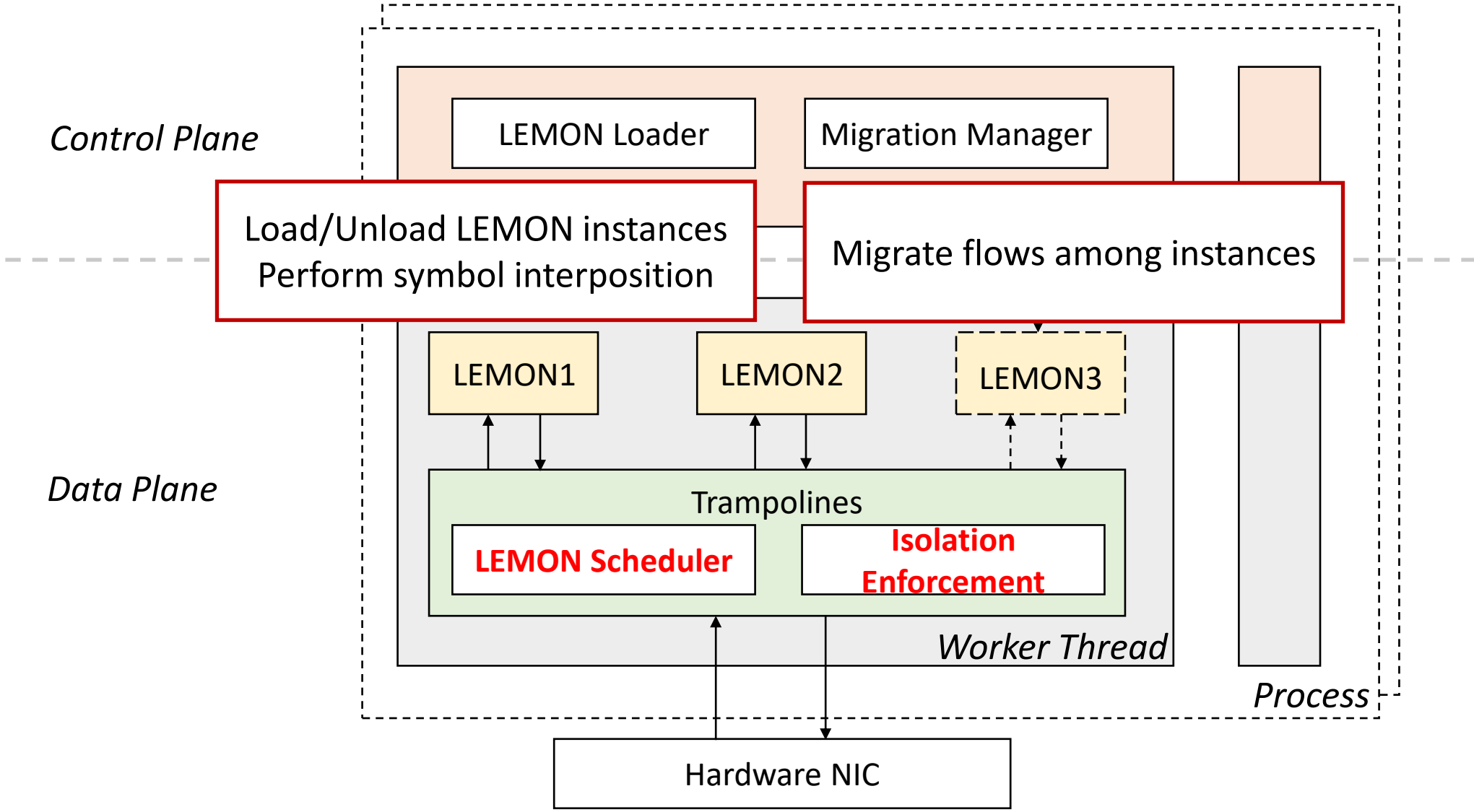
LemonNFV Overview



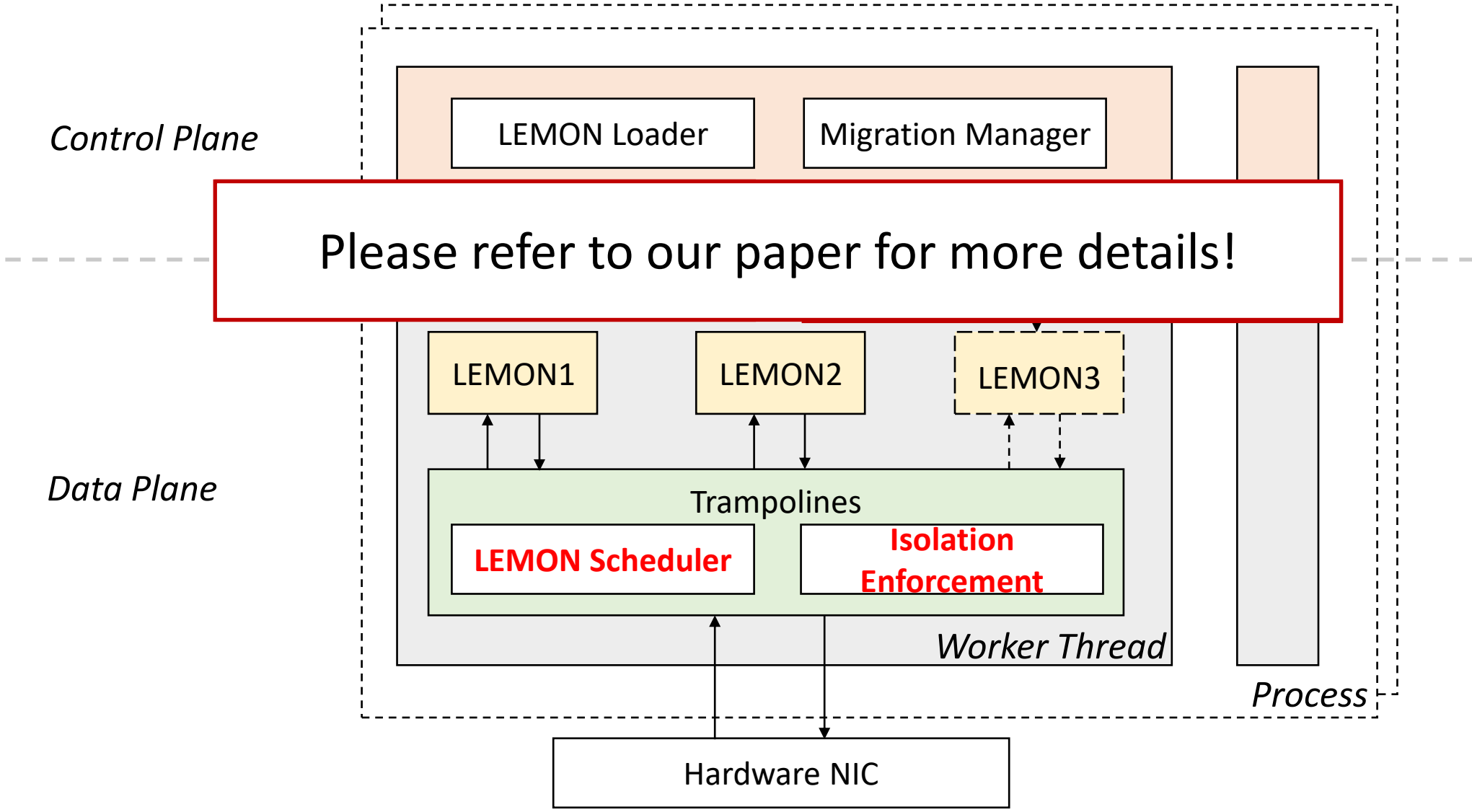
LemonNFV Overview



LemonNFV Overview



LemonNFV Overview

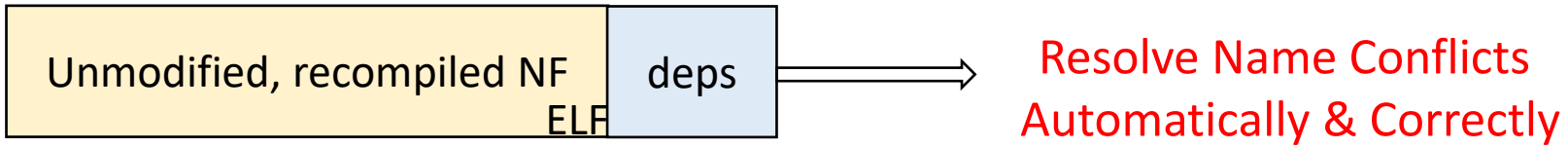


The LEMON Abstraction
(LEast Modified network functiON)

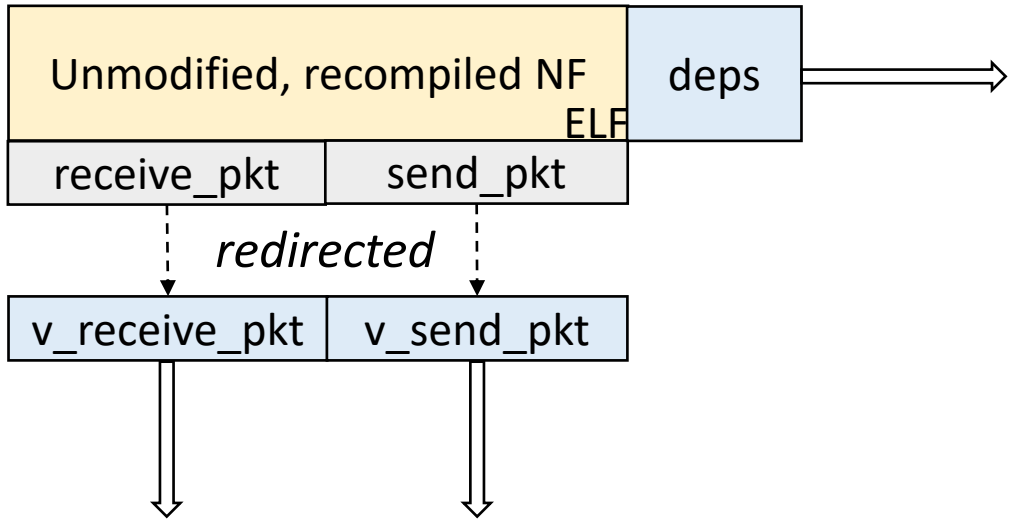
The LEMON Abstraction
(LEast Modified network functiON)

Unmodified, recompiled NF ELF

The LEMON Abstraction
(LEast Modified network functiON)



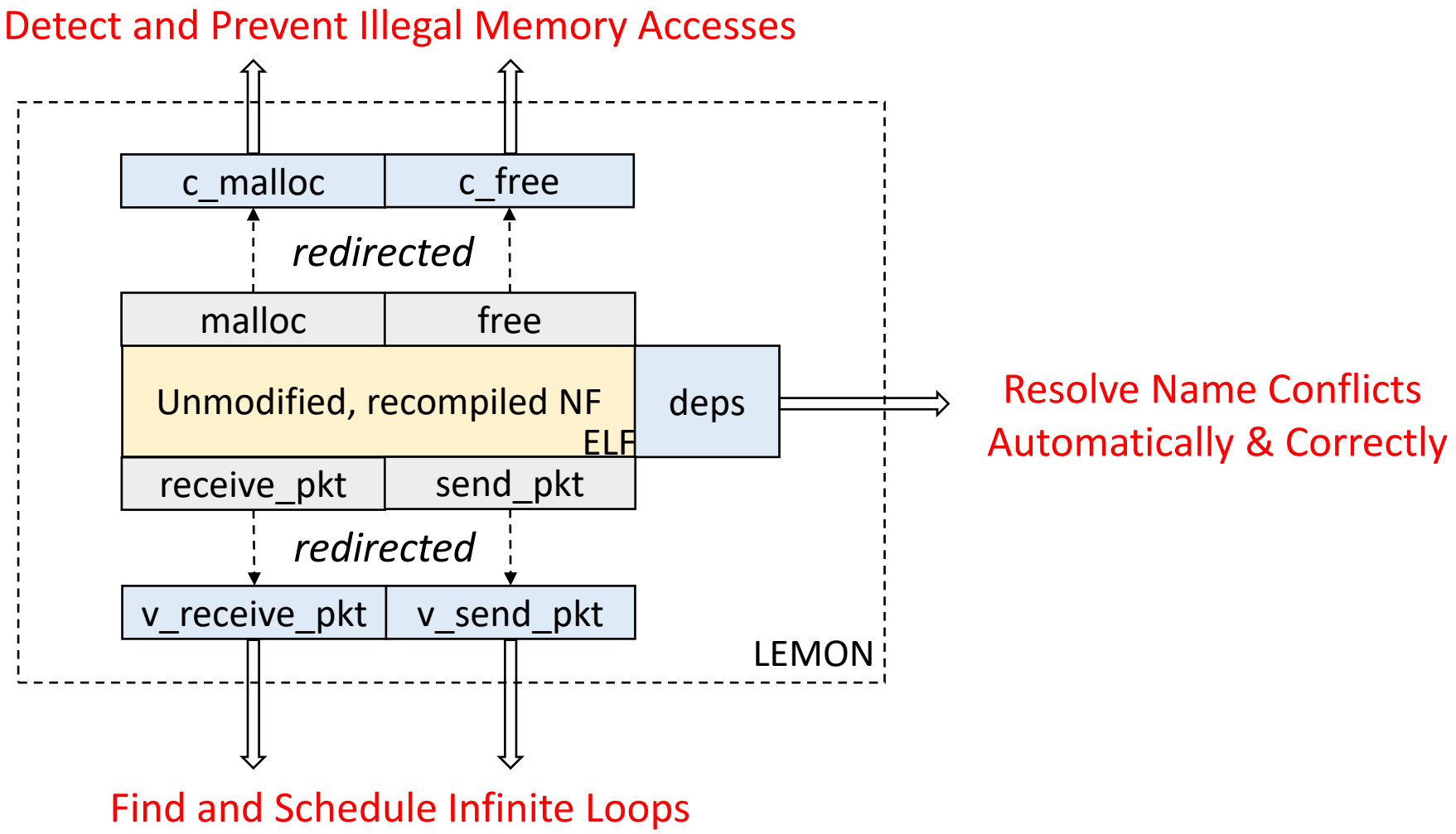
The LEMON Abstraction
(LEast Modified network functiON)



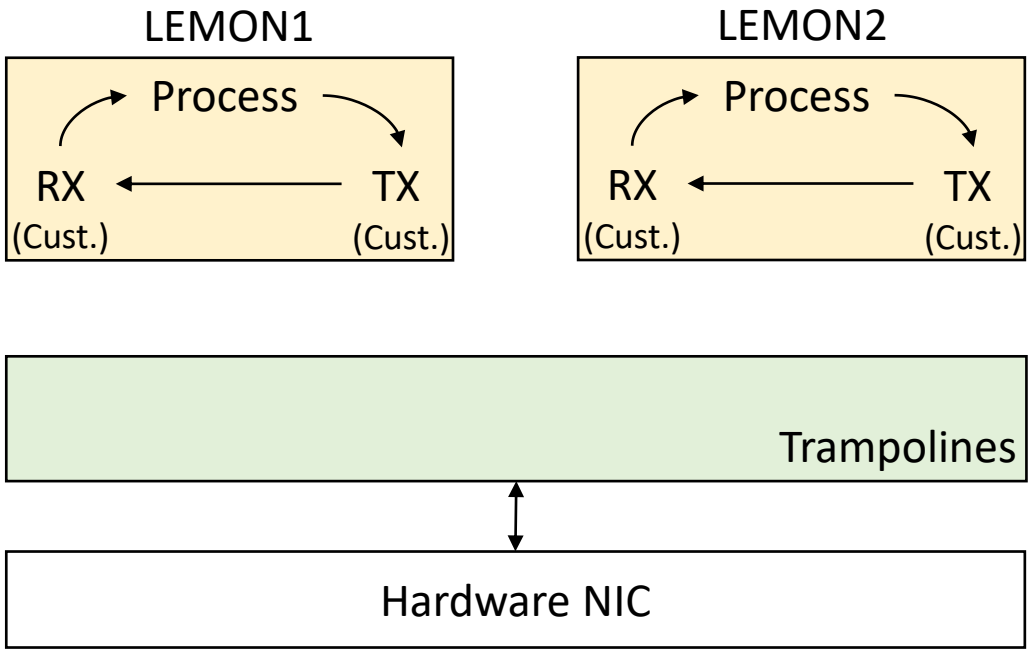
Resolve Name Conflicts
Automatically & Correctly

Find and Schedule Infinite Loops

The LEMON Abstraction (LEast Modified network functiON)

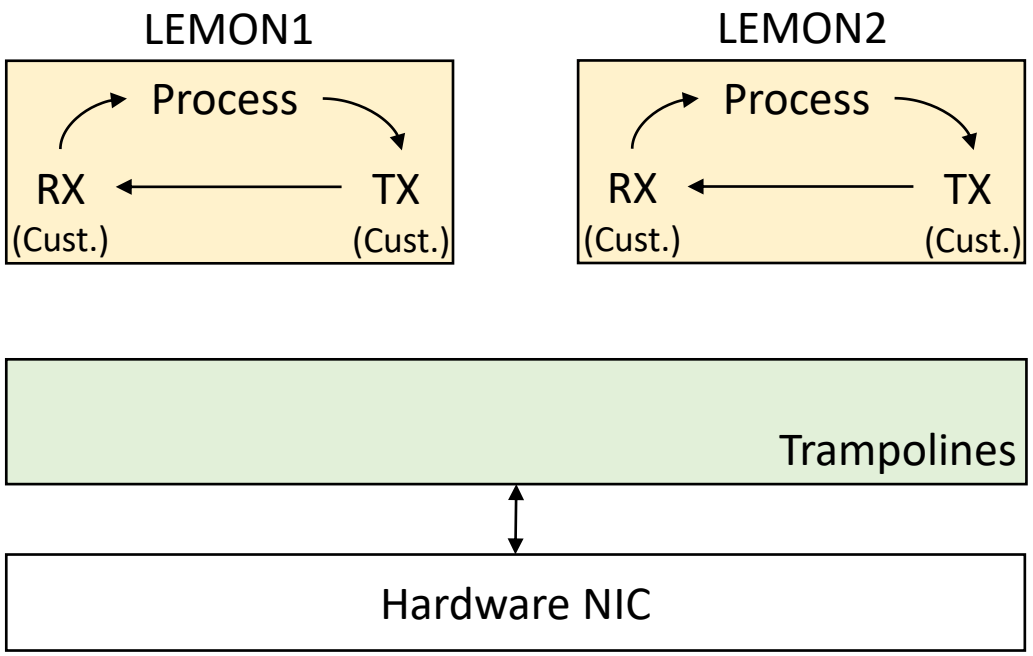


Scheduling the LEMONs with Customized I/O



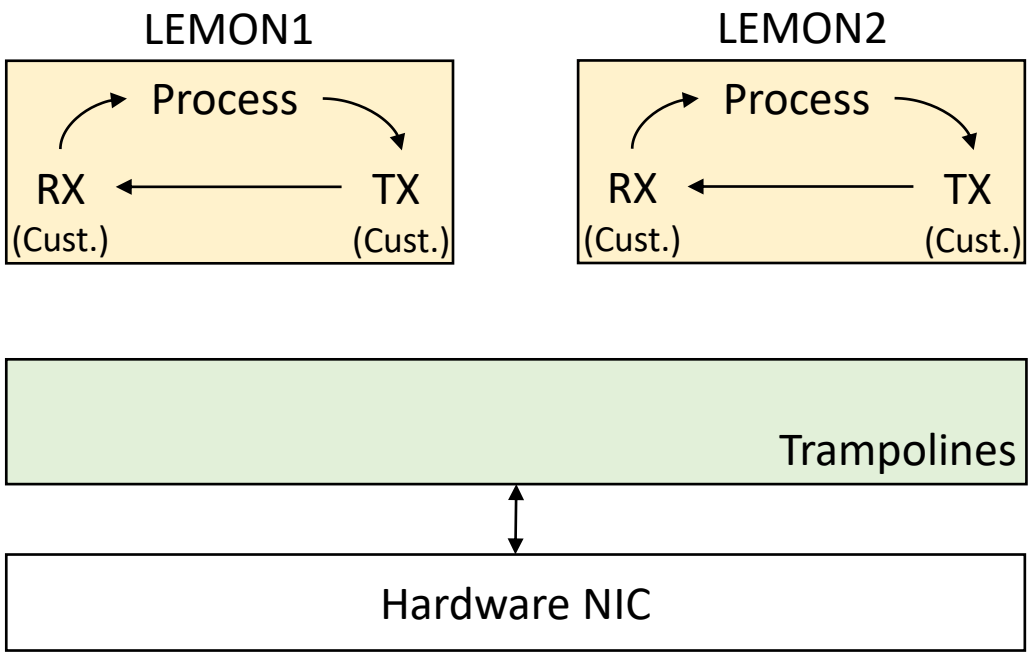
- By default, NFs process packets in an infinite loop
  - RX/TX talks directly to NIC

Scheduling the LEMONs with Customized I/O



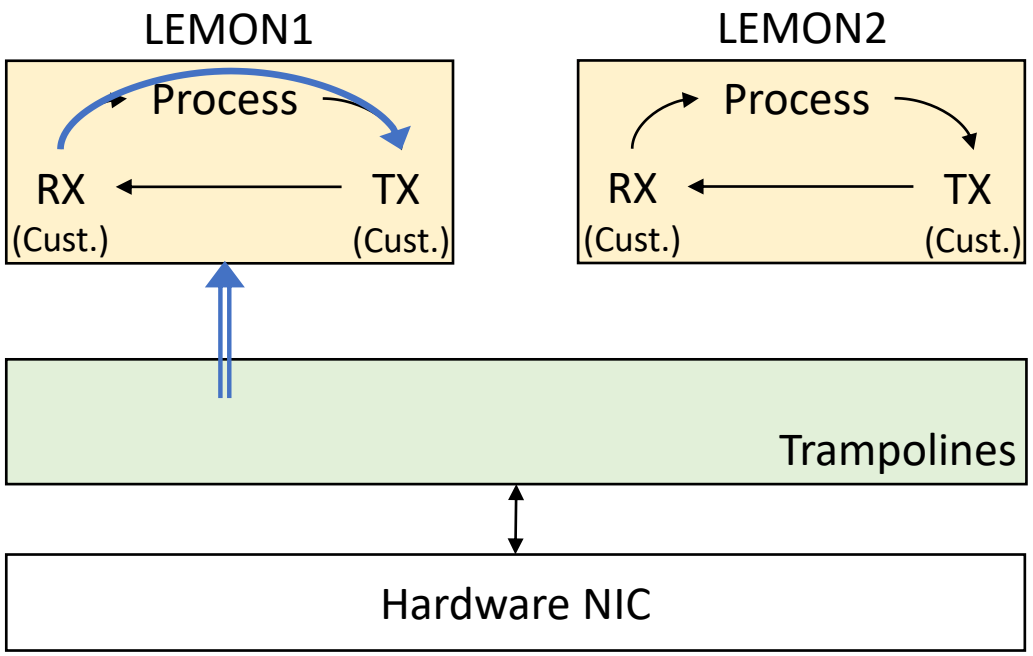
- By default, NFs process packets in an infinite loop
  - RX/TX talks directly to NIC
- Customized I/O does not modify the loop but schedules it

Scheduling the LEMONs with Customized I/O



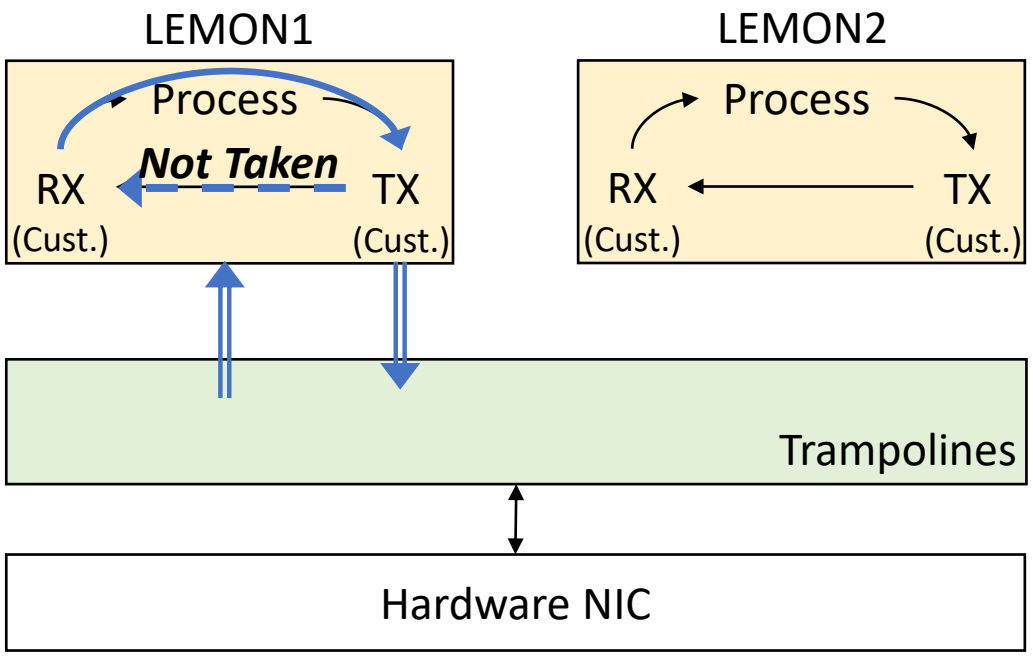
- By default, NFs process packets in an infinite loop
  - RX/TX talks directly to NIC
- Customized I/O does not modify the loop but schedules it
  - Using RX/TX as scheduling points
  - Calling TX saves context and returns to the trampolines
  - The trampolines select the next LEMON and restore its (post-TX) context

Scheduling the LEMONs with Customized I/O



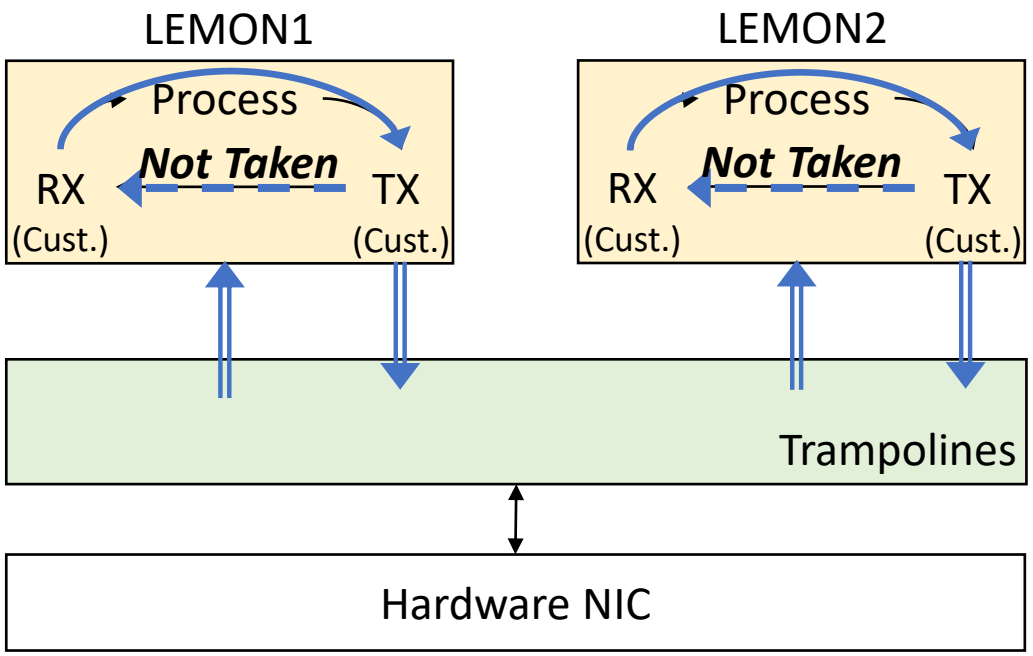
- By default, NFs process packets in an infinite loop
  - RX/TX talks directly to NIC
- Customized I/O does not modify the loop but schedules it
  - Using RX/TX as scheduling points
  - Calling TX saves context and returns to the trampolines
  - The trampolines select the next LEMON and restore its (post-TX) context

Scheduling the LEMONs with Customized I/O



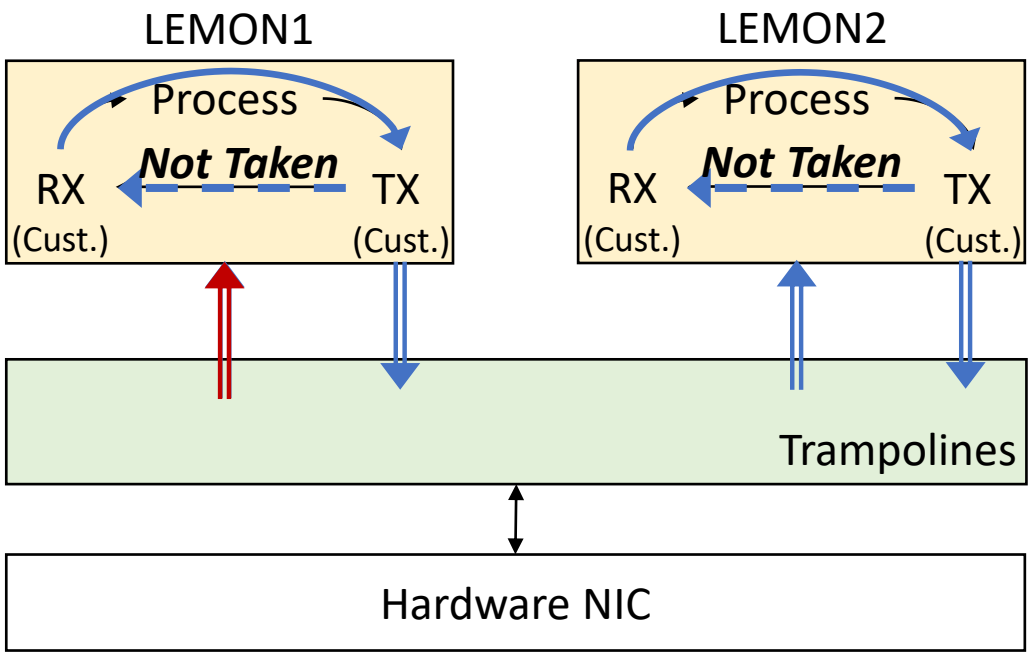
- By default, NFs process packets in an infinite loop
  - RX/TX talks directly to NIC
- Customized I/O does not modify the loop but schedules it
  - Using RX/TX as scheduling points
  - Calling TX saves context and returns to the trampolines
  - The trampolines select the next LEMON and restore its (post-TX) context

Scheduling the LEMONs with Customized I/O



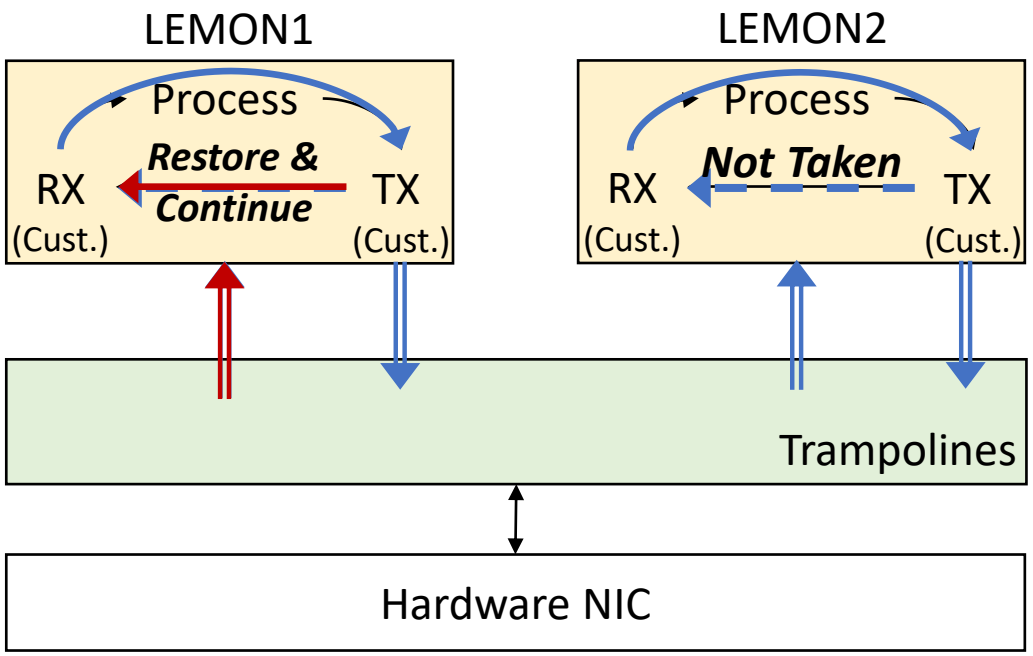
- By default, NFs process packets in an infinite loop
  - RX/TX talks directly to NIC
- Customized I/O does not modify the loop but schedules it
  - Using RX/TX as scheduling points
  - Calling TX saves context and returns to the trampolines
  - The trampolines select the next LEMON and restore its (post-TX) context

Scheduling the LEMONs with Customized I/O



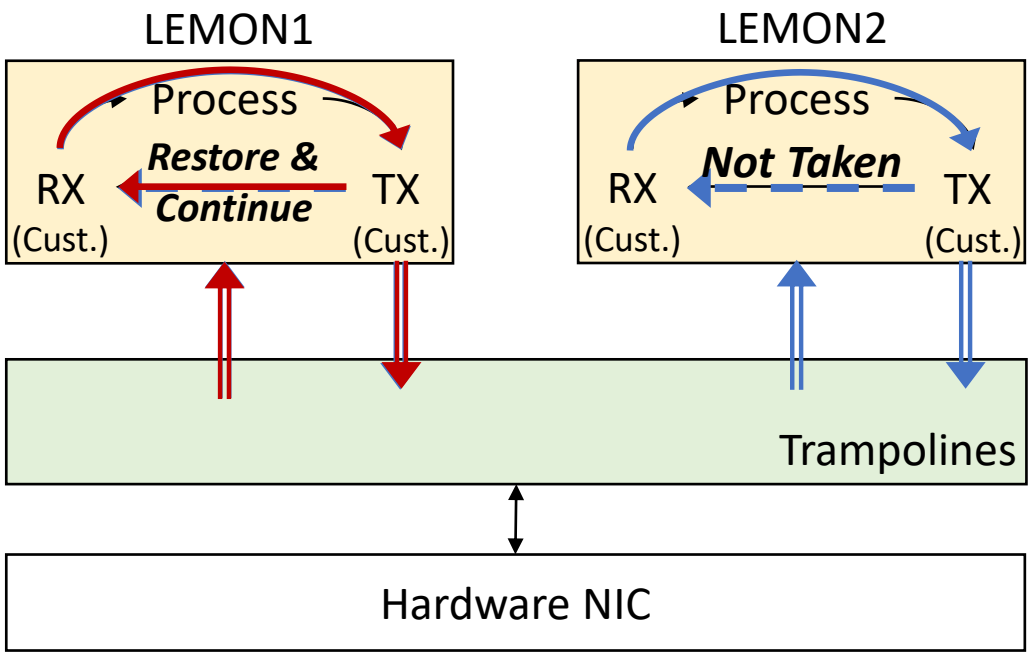
- By default, NFs process packets in an infinite loop
  - RX/TX talks directly to NIC
- Customized I/O does not modify the loop but schedules it
  - Using RX/TX as scheduling points
  - Calling TX saves context and returns to the trampolines
  - The trampolines select the next LEMON and restore its (post-TX) context

Scheduling the LEMONs with Customized I/O



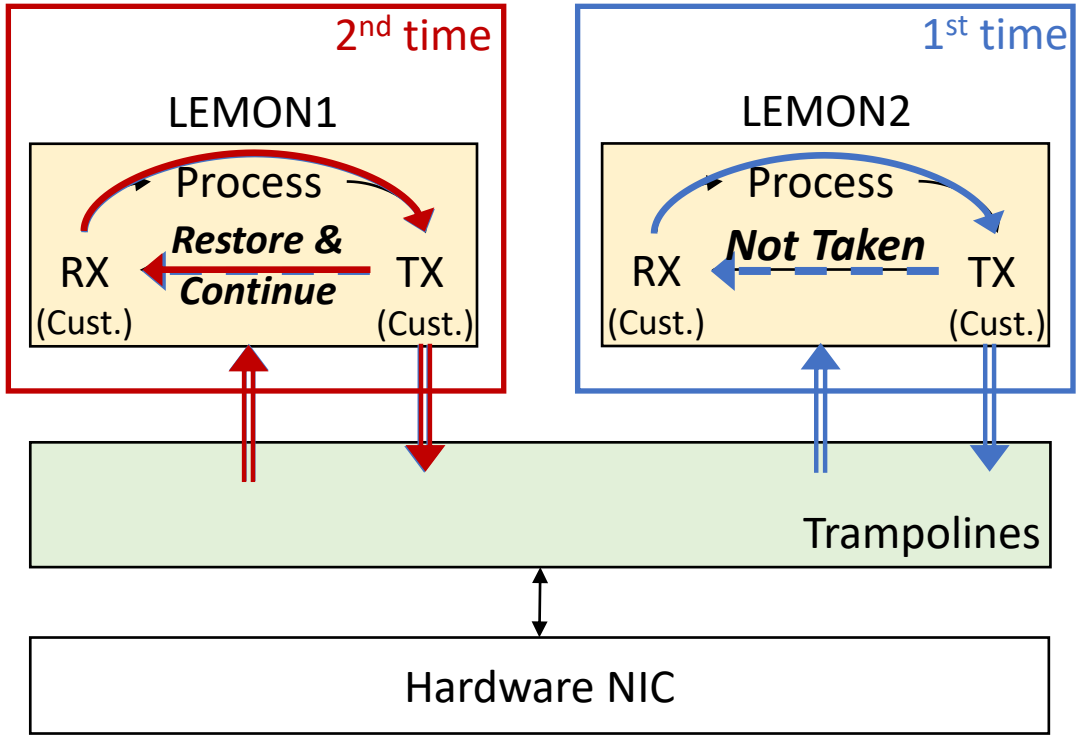
- By default, NFs process packets in an infinite loop
  - RX/TX talks directly to NIC
- Customized I/O does not modify the loop but schedules it
  - Using RX/TX as scheduling points
  - Calling TX saves context and returns to the trampolines
  - The trampolines select the next LEMON and restore its (post-TX) context

Scheduling the LEMONs with Customized I/O



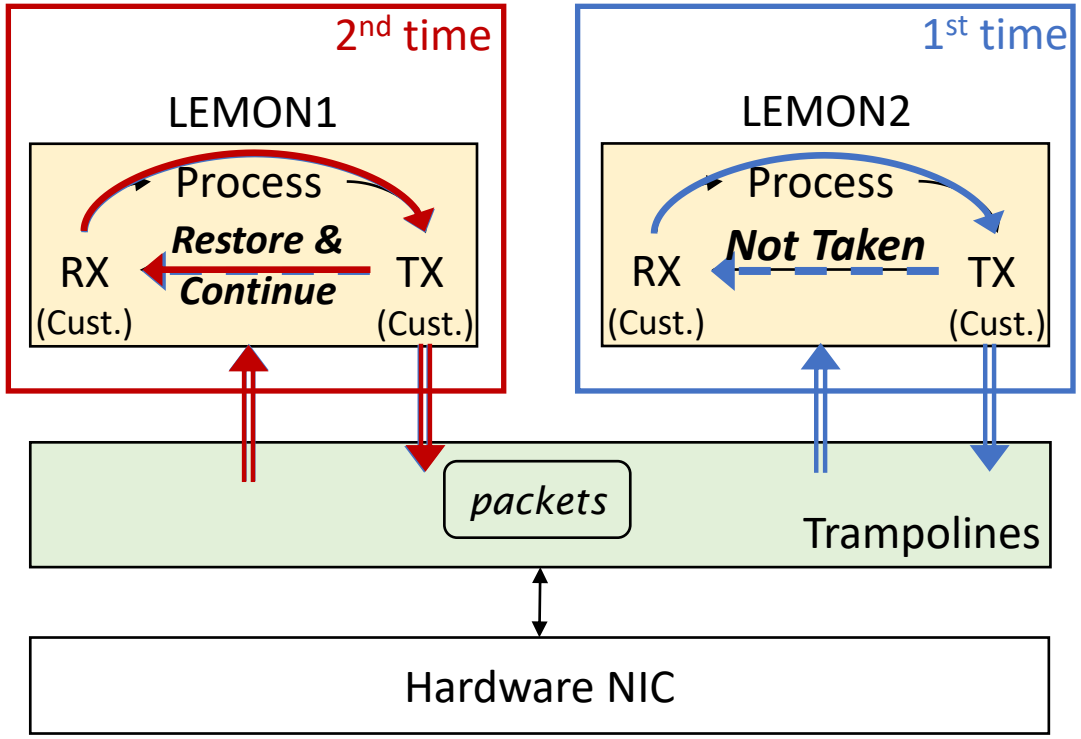
- By default, NFs process packets in an infinite loop
  - RX/TX talks directly to NIC
- Customized I/O does not modify the loop but schedules it
  - Using RX/TX as scheduling points
  - Calling TX saves context and returns to the trampolines
  - The trampolines select the next LEMON and restore its (post-TX) context

Scheduling the LEMONs with Customized I/O



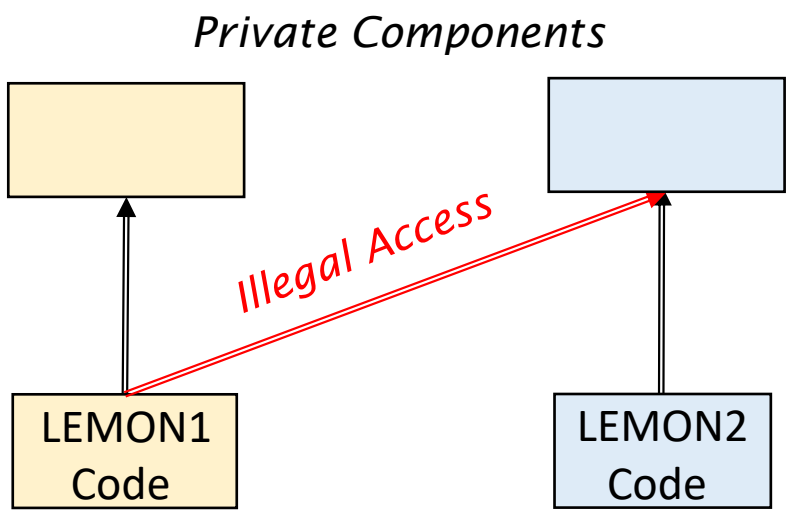
- By default, NFs process packets in an infinite loop
  - RX/TX talks directly to NIC
- Customized I/O does not modify the loop but schedules it
  - Using RX/TX as scheduling points
  - Calling TX saves context and returns to the trampolines
  - The trampolines select the next LEMON and restore its (post-TX) context

Scheduling the LEMONs with Customized I/O



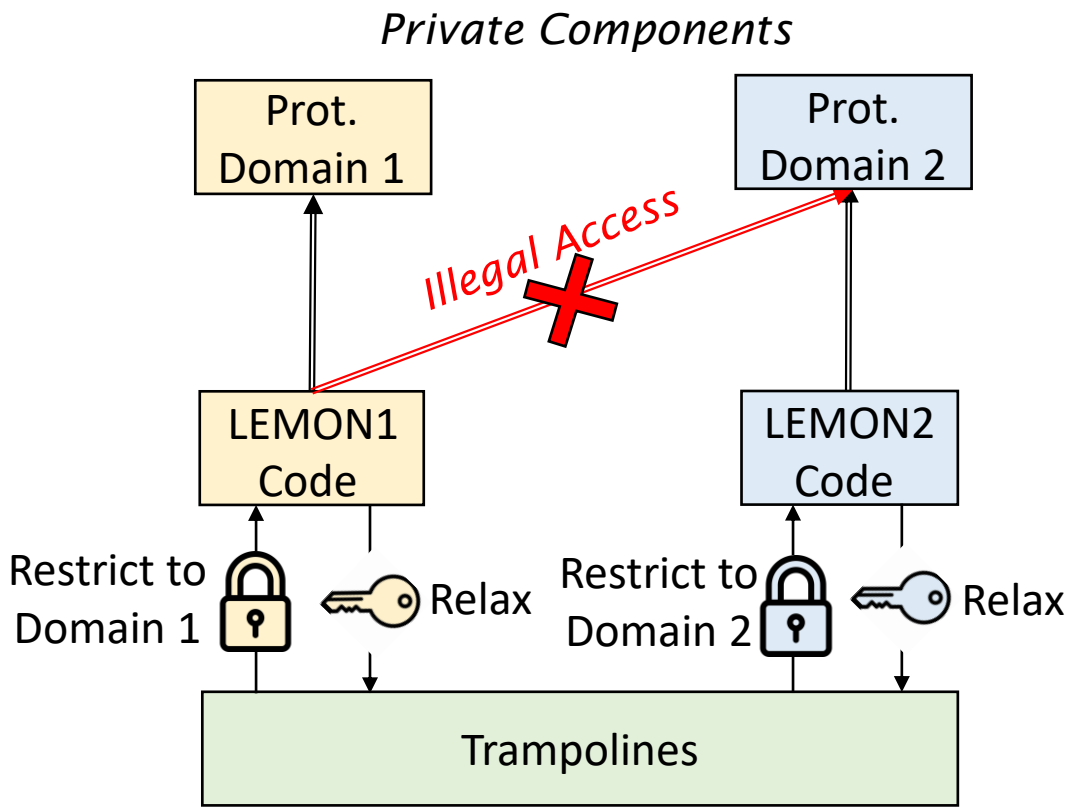
- By default, NFs process packets in an infinite loop
  - RX/TX talks directly to NIC
- Customized I/O does not modify the loop but schedules it
  - Using RX/TX as scheduling points
  - Calling TX saves context and returns to the trampolines
  - The trampolines select the next LEMON and restore its (post-TX) context
  - All LEMONS access packets from buffer inside the trampolines

Preventing Illegal Memory Accesses



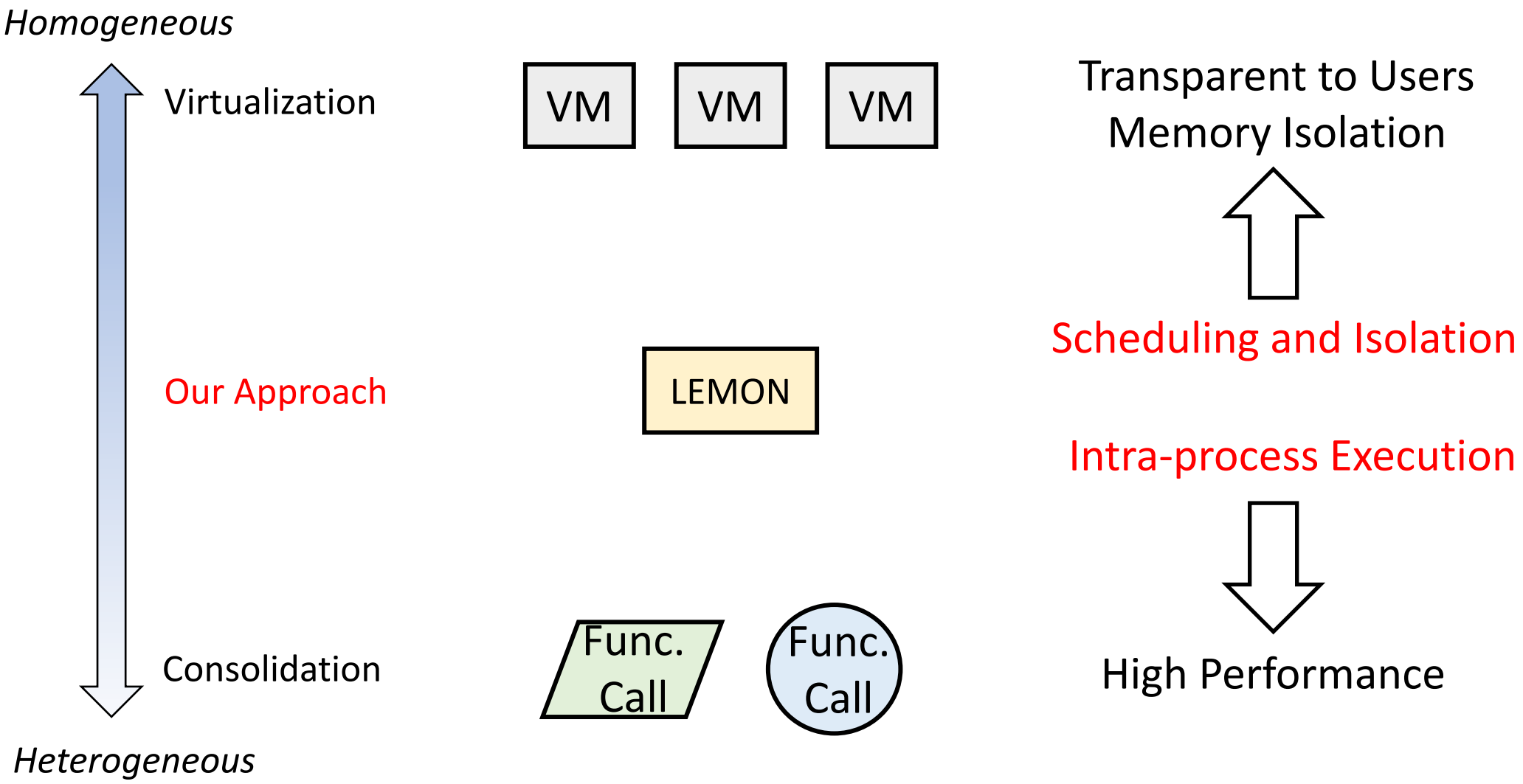
- The design of LEMON creates bounded memory regions
  - Private heap, stack and dependencies instead of shared ones
  - Accesses outside its own region is illegal

Preventing Illegal Memory Accesses



- The design of LEMON creates bounded memory regions
  - Private heap, stack and dependencies instead of shared ones
  - Accesses outside its own region is illegal
- Bounded memory is efficiently isolated by domain switching
  - LemonNFV uses Intel® Protection Key for Userspace (PKU)
  - Restrict access before switching to LEMONs, and relax it before switching back to trampolines

Design Takeaway



Evaluation

- Effort of LemonNFV to consolidate heterogeneous NFs

- Performance compared with State-Of-The-Art NFV systems

Minimum LOC Modification to Interoperation

Heterogeneity of Real World NFs

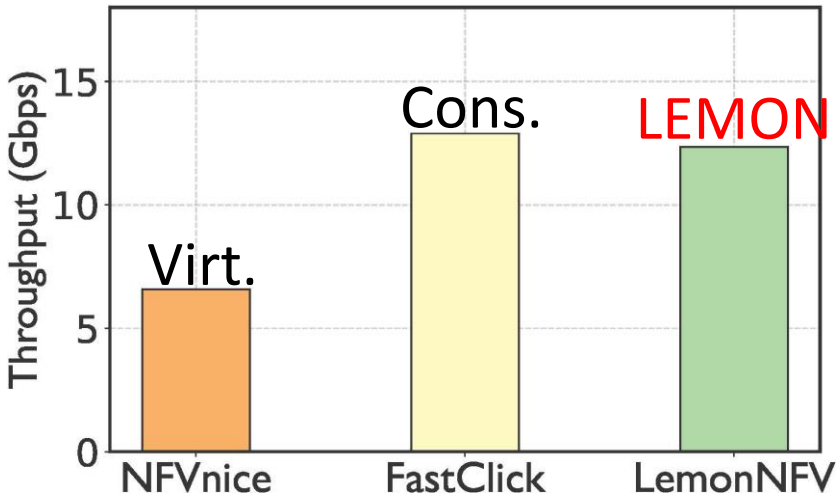
Huge Code Base Of Real World NFs

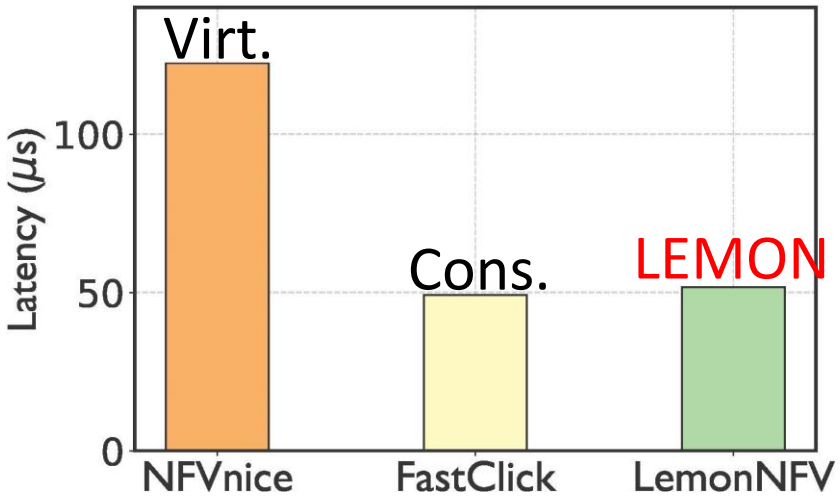
Effort of LemonNFV

NF	Framework	Language	I/O	NF LOC	Framework LOC	Modified LOC
IDS	Rubik	C	DPDK	337	31K	2
NAT	FastClick	C++	DPDK	94	331K	2
ACL	NetBricks	Rust	DPDK	401	58K	8
CT	mOS	C	libpcap	325	139K	4
DPI	nDPI	C	libpcap	4498	121K	2

LemonNFV consolidates heterogeneous NFs without much effort (LOC)

Comparing Performance with State-Of-The-Art



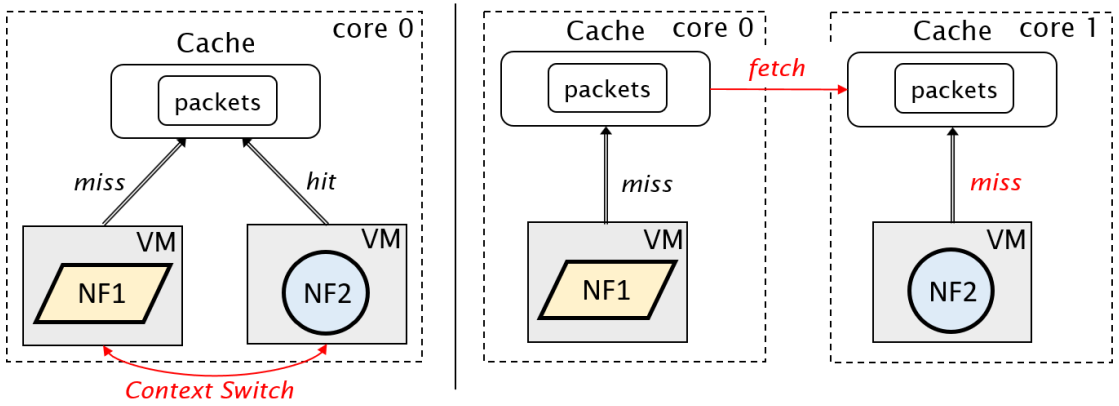


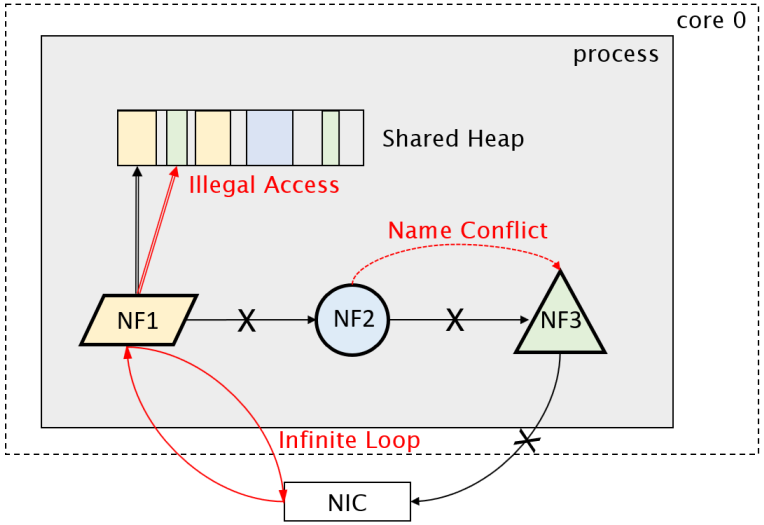
NFVnice: SOTA in virtualization +88% throughput, -58% latency

FastClick: SOTA in consolidation -4.1% throughput, +4.9% latency

LemonNFV consolidates heterogeneous NFs with minor overhead

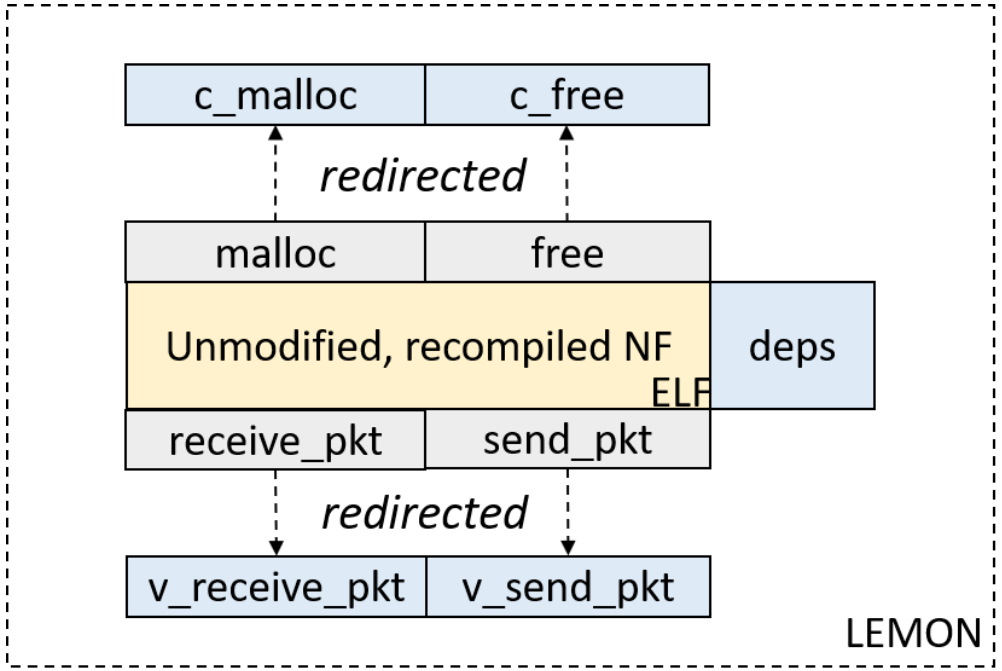
Summary





- Virtualization nor direct consolidation achieves heterogeneous NF interoperation
  - Virtualization overhead
  - Effort of code modification

Summary



- Virtualization nor direct consolidation achieves heterogeneous NF interoperation
  - Virtualization overhead
  - Effort of code modification
- LemonNFV consolidates NFs with minor overhead and effort
  - Designs a unique abstraction LEMON
  - Schedules and isolates LEMONs inside one process

Please Read Our Paper for More Details!

LemonNFV: Consolidating Heterogeneous Network Functions at Line Speed

Hao Li[1], Yihan Dang[1], Guangda Sun[1,2], Guyue Liu[3], Danfeng Shan[1], Peng Zhang[1]

[1]*Xi'an Jiaotong University* [2]*National University of Singapore* [3]*New York University Shanghai*

Yihan Dang

yhdang@stu.xjtu.edu.cn