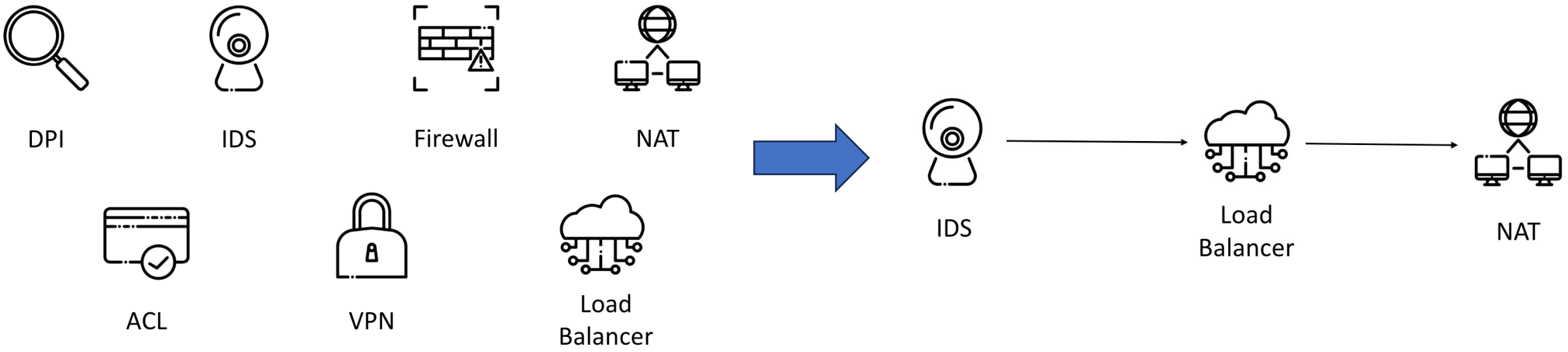
LemonNFV: Consolidating Heterogeneous Network Functions at Line Speed

Hao Li, Yihan Dang et. al. Xian Jiaotong University

NSDI23

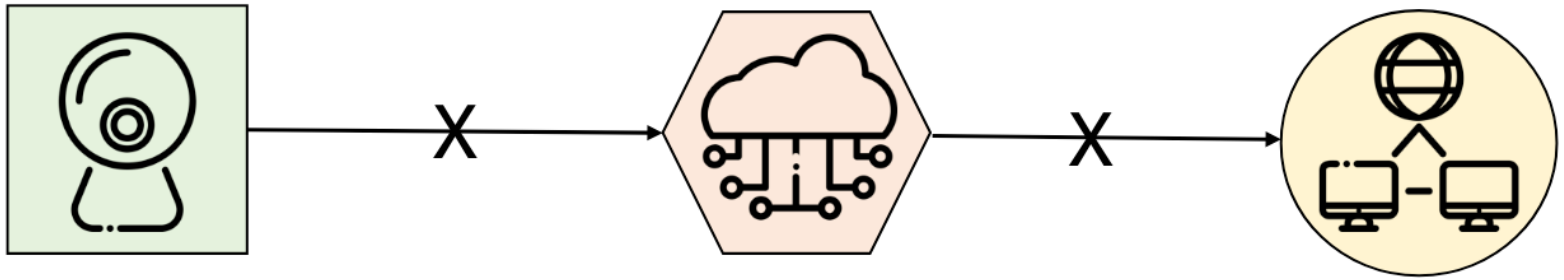
VNF and SFC

- Virtualized Network Function (VNF)
- Service Function Chain (SFC)



Difficulties in NFs' Interoperation

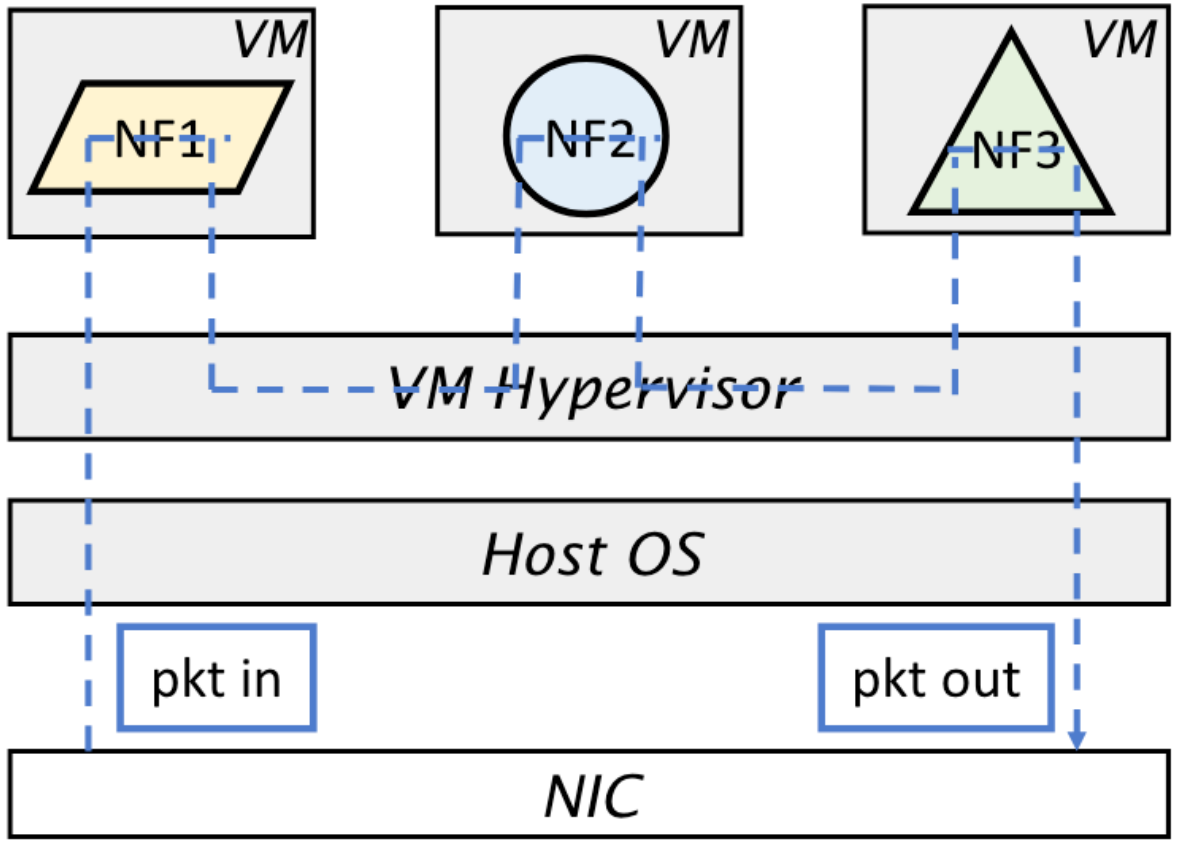
- **Heterogeneous NFs Are Not Interoperable**
  - Programming language
  - Execution model
  - State & Packet Abstraction



How can heterogeneous NFs interoperate?

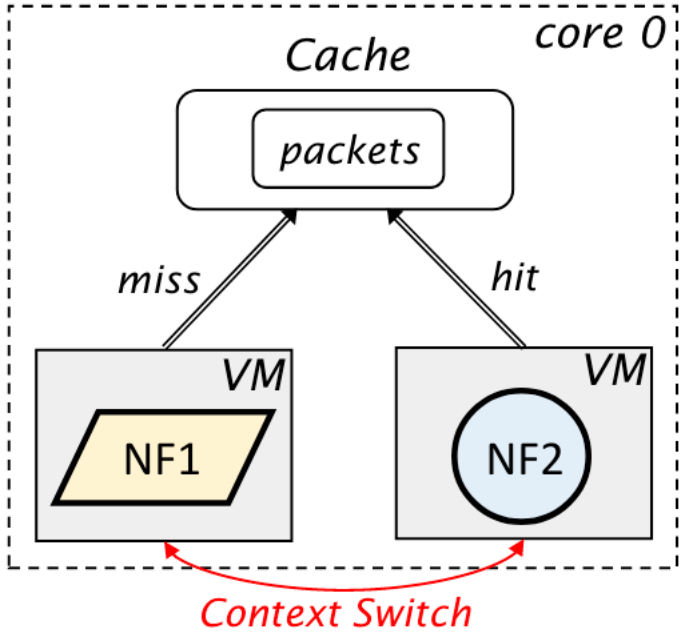
Existing Solutions

- Solution 1: Virtualization

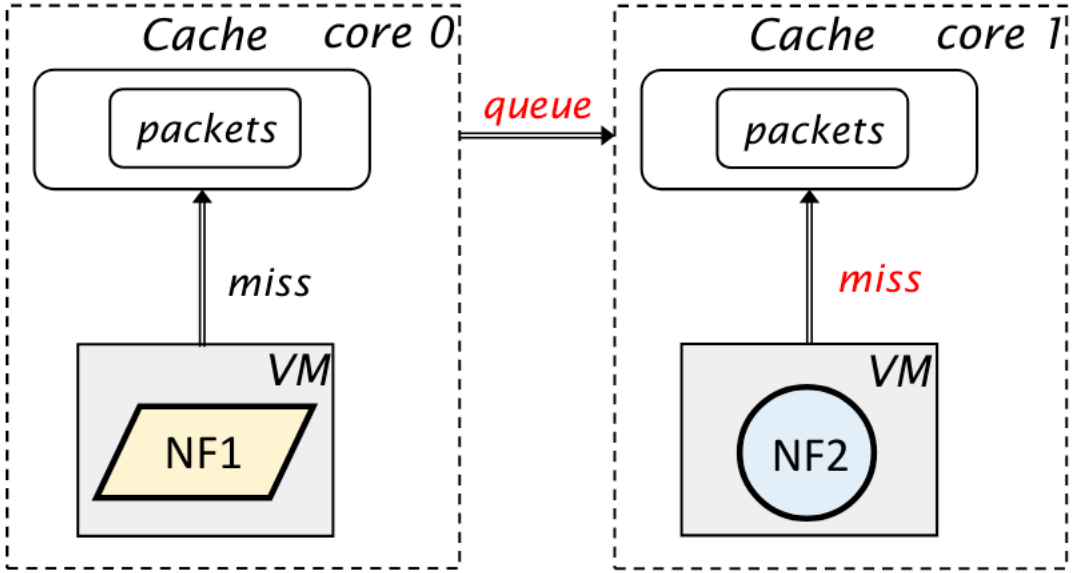


Existing Solutions

- Dilemma: Context Switch vs Inter-Core Traffic



- Scheduling instances on the same core

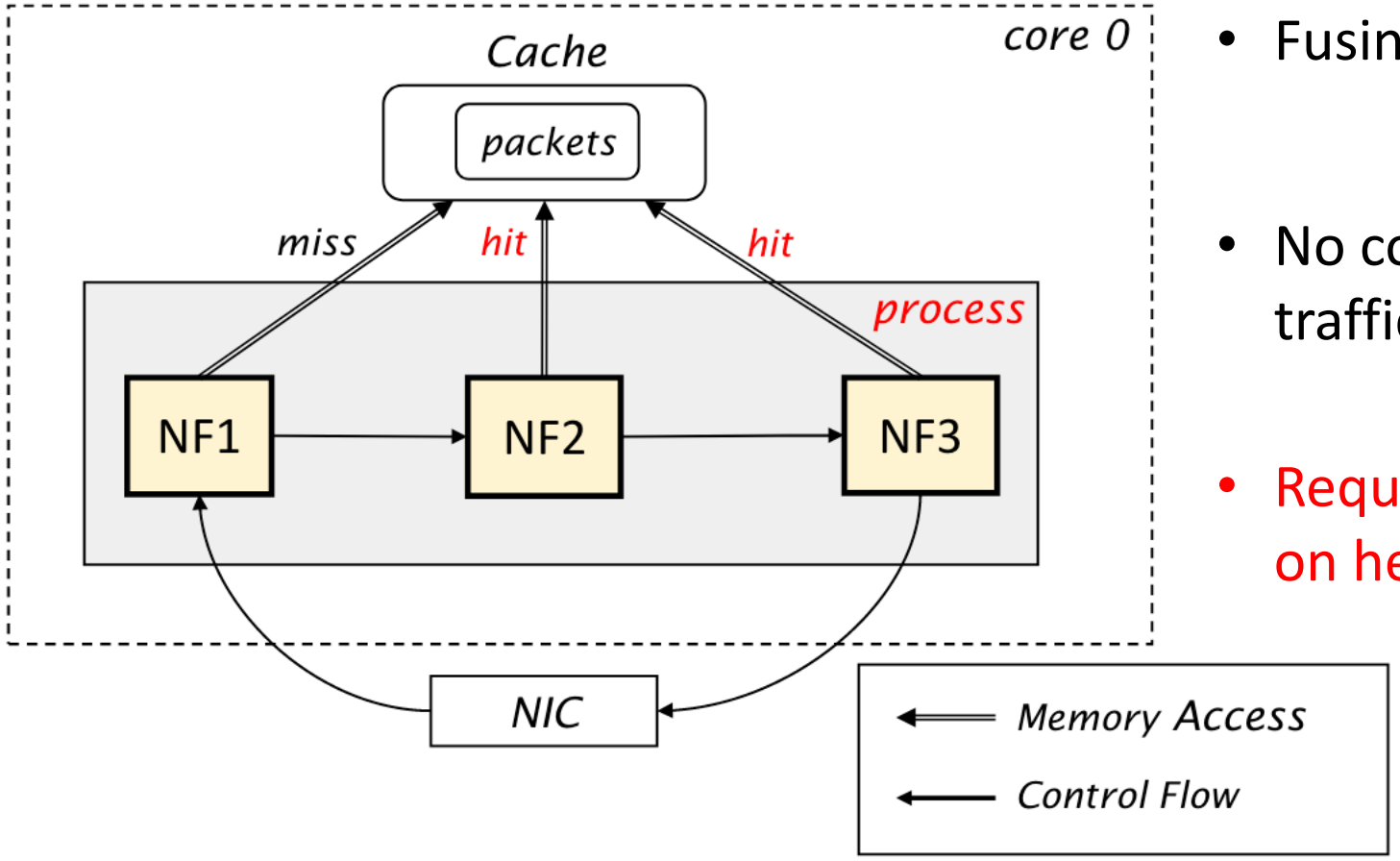


- Pinning instances on dedicated cores

- Virtualization approaches are hard to reach line rate

Existing Solutions

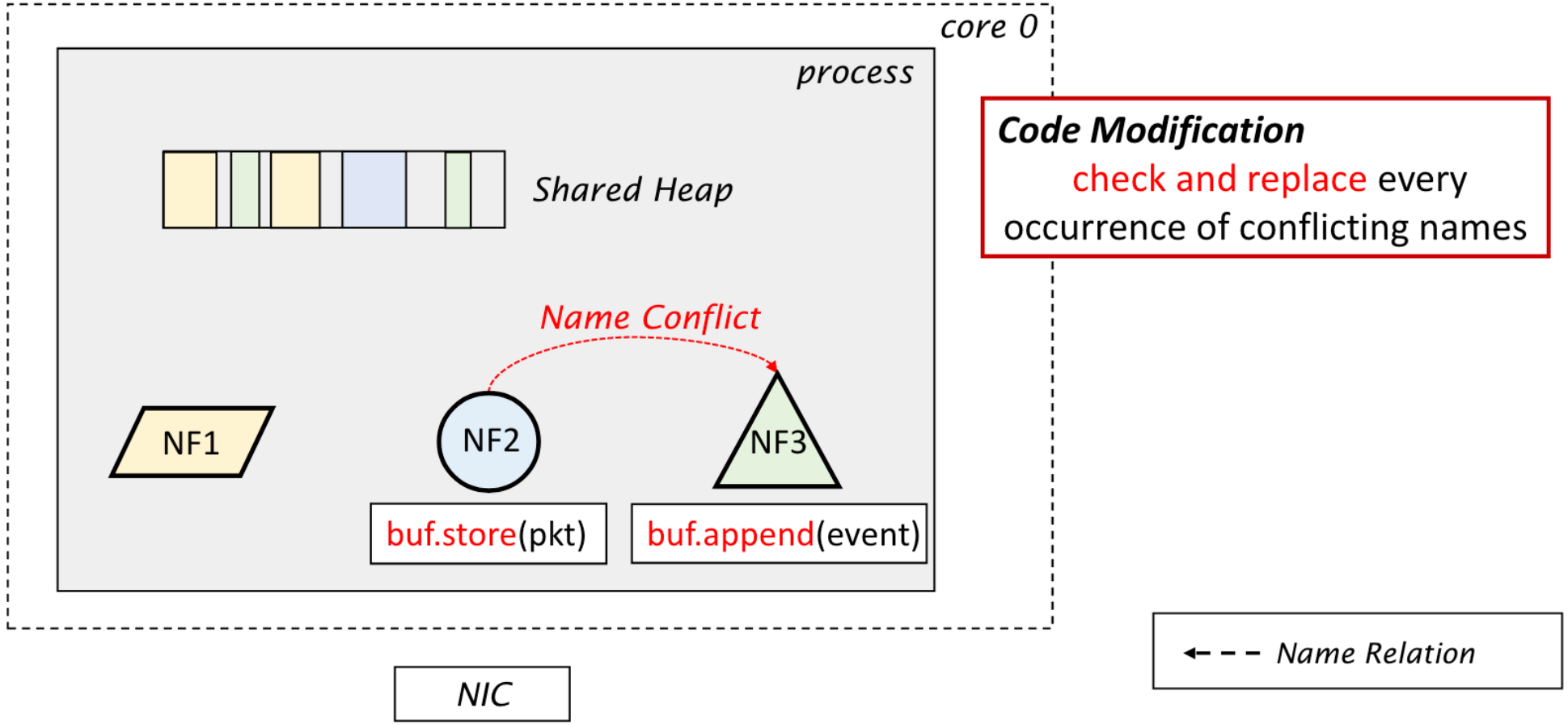
- Solution 2: Consolidation



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

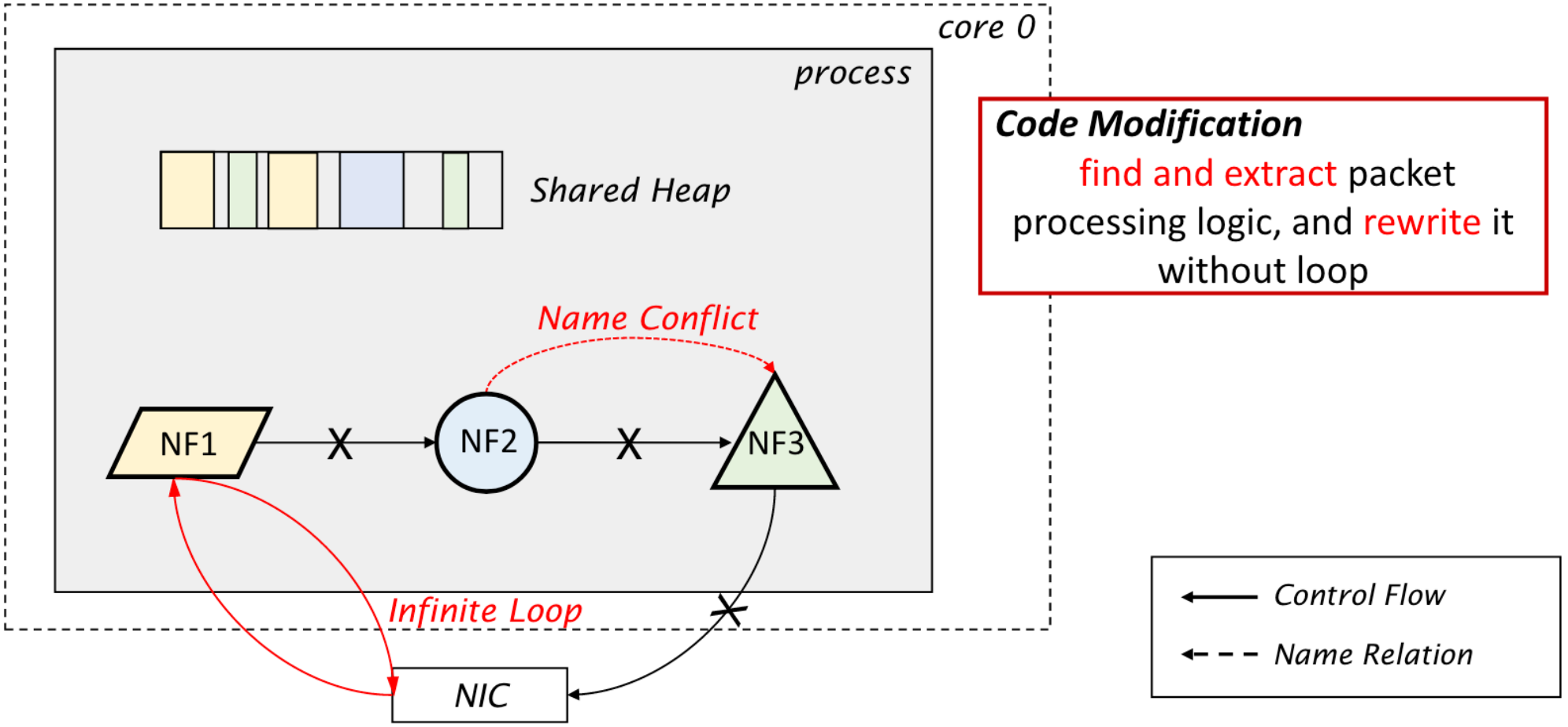
Existing Solutions

- Problem 1: Namespace Conflict



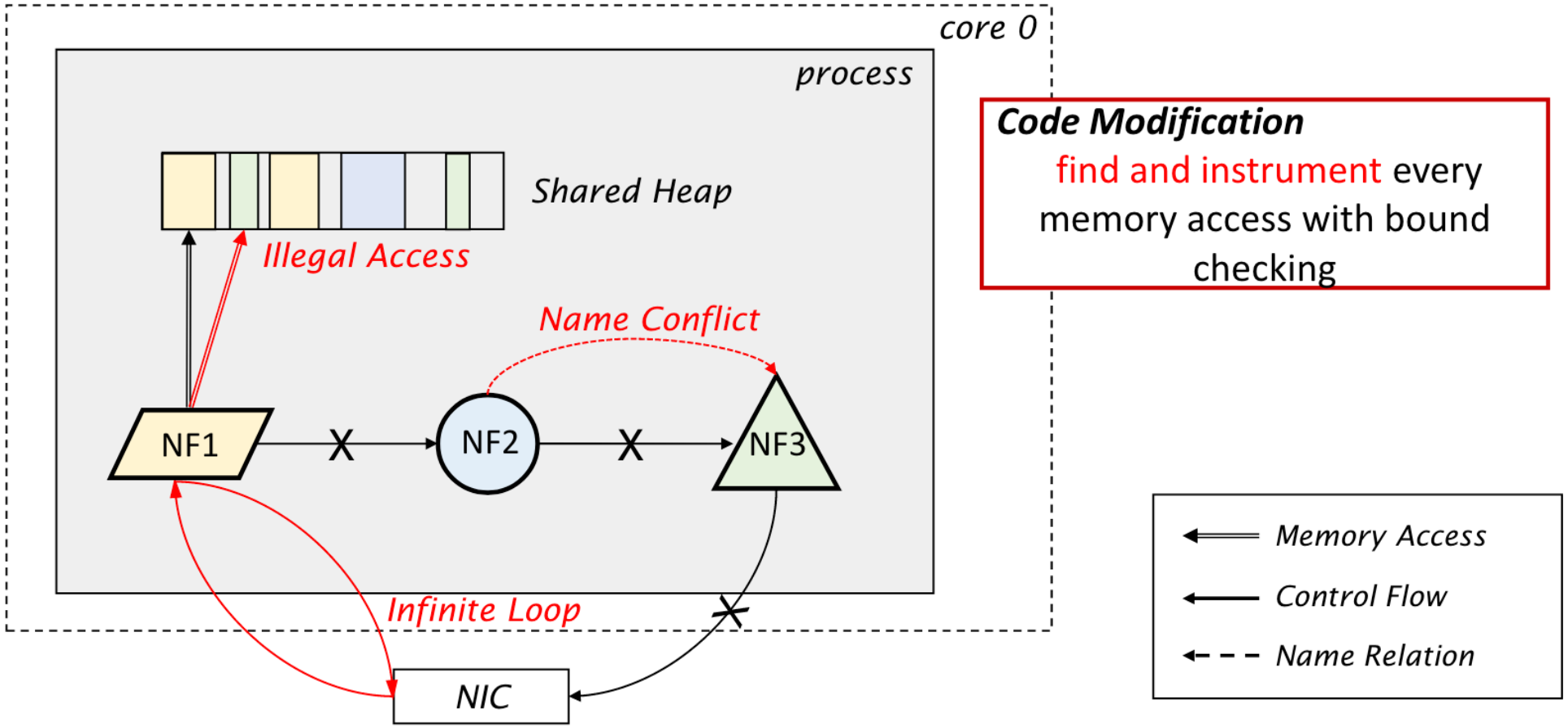
Existing Solutions

- Problem 2: Private Control Flow



Existing Solutions

- Problem 3: Illegal Memory Access



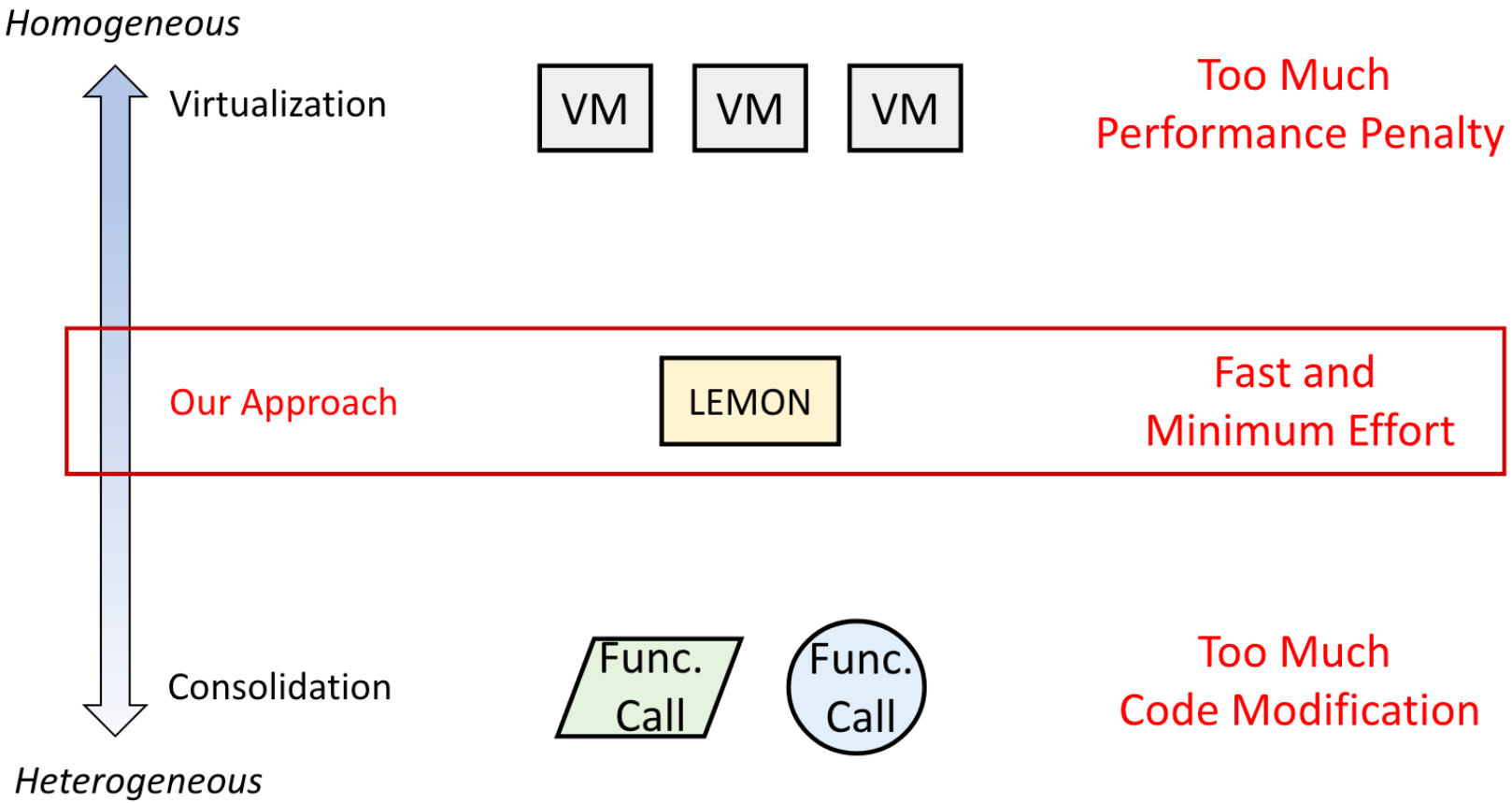
Takeaways on Existing Solutions

- Solution 1: Virtualization
  - Good: No modification to NFs' codes
  - Bad: Virtualization approaches are hard to reach line rate!

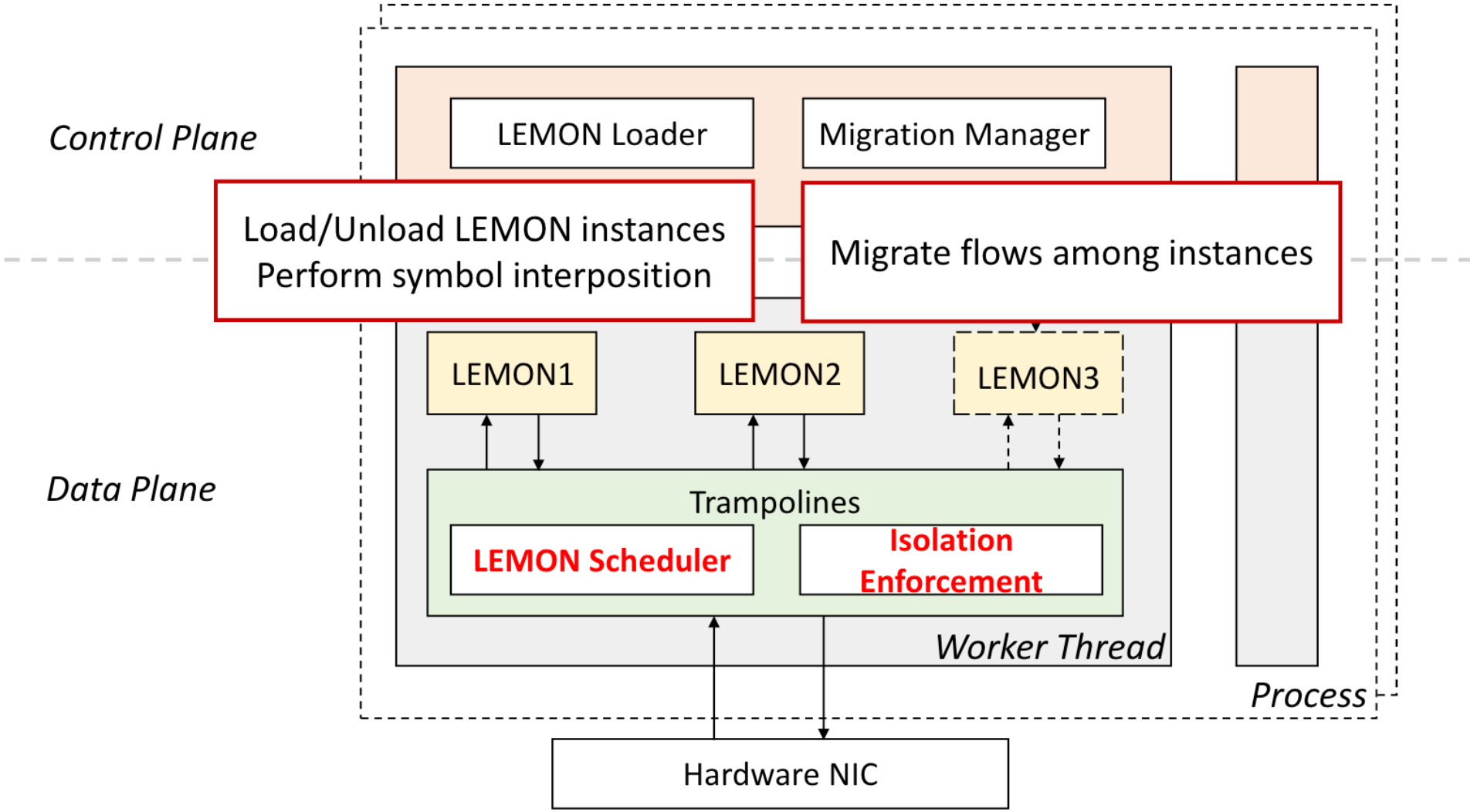
- Solution 2: Consolidation
  - Good: High performance
  - Bad: Direct Consolidation Forces Huge Code Modification!

LemonNFV's Solution

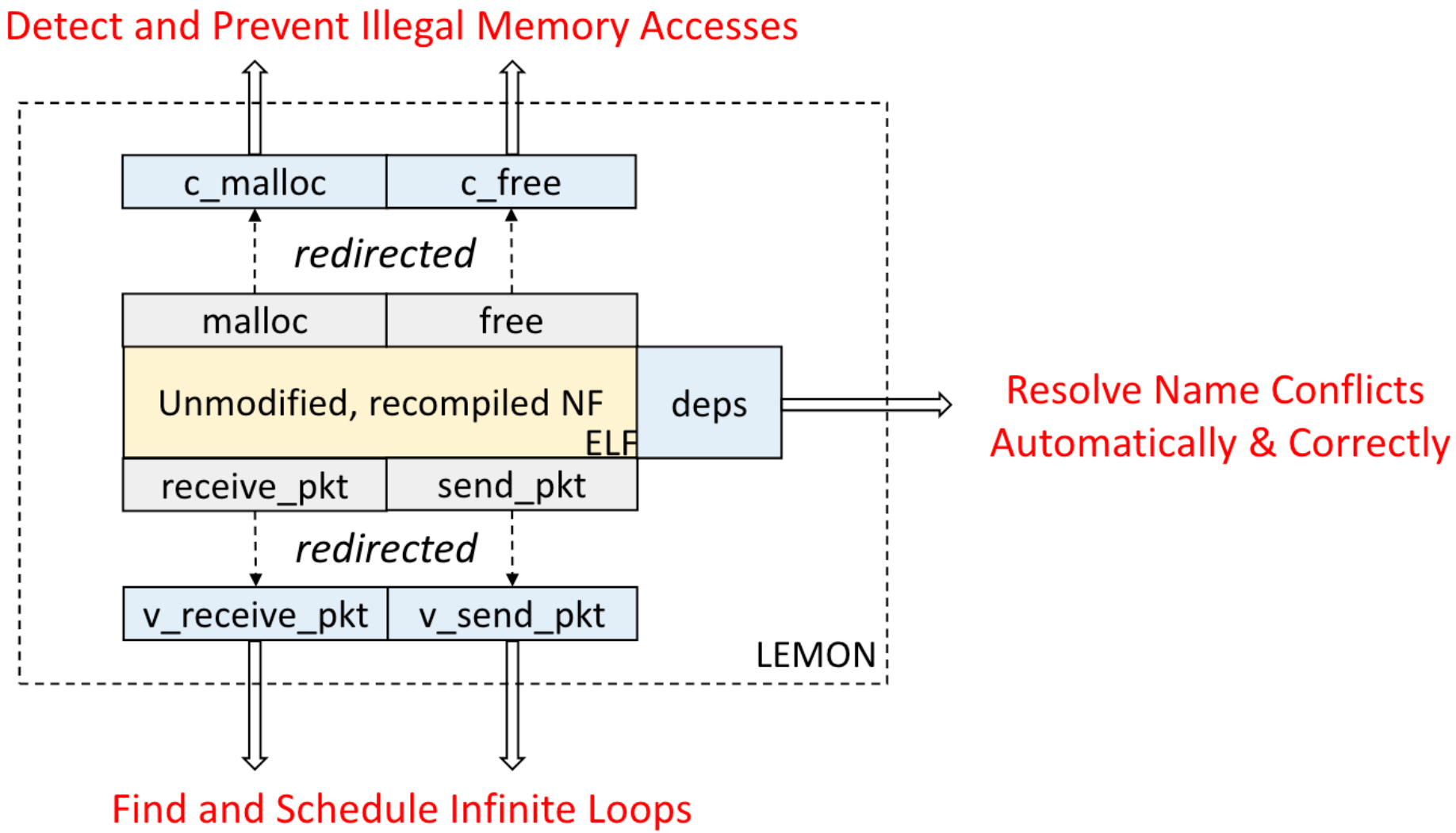
- Insights: exploit the advantages of the two methods



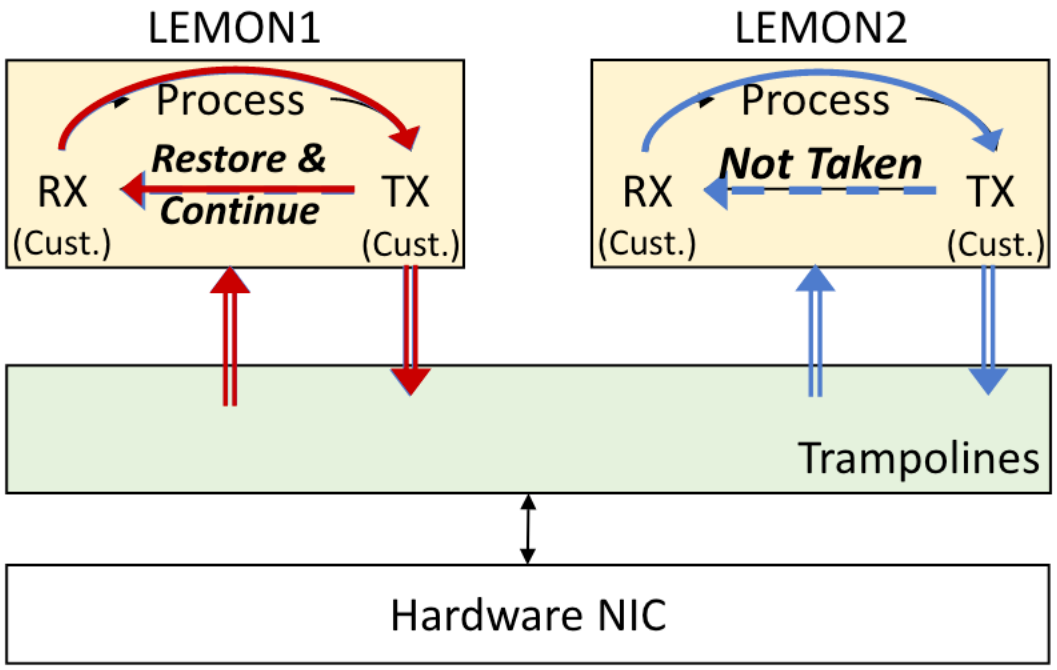
LemonNFV Overview



The LEMON Abstraction (LEast Modified network functiON)

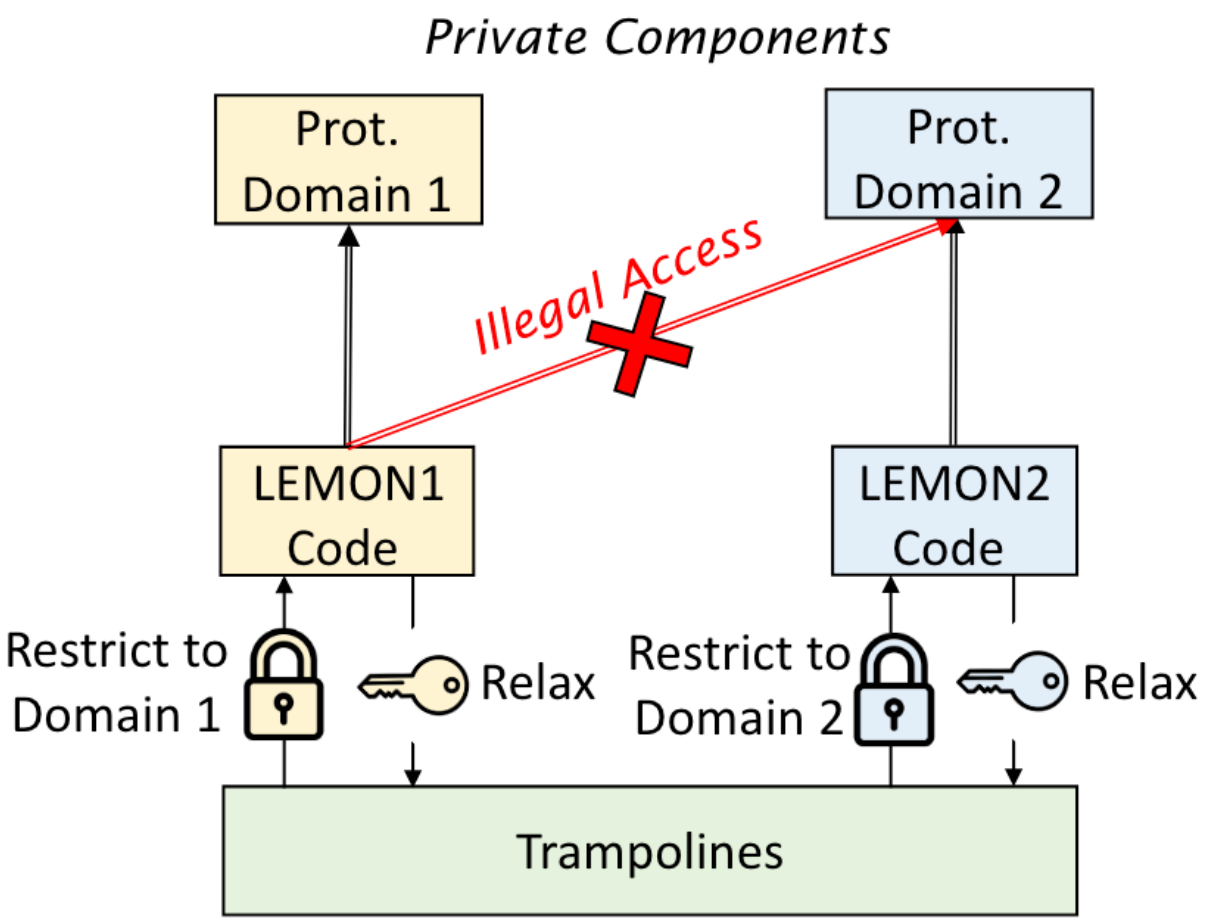


Scheduling the LEMONs with Customized I/O



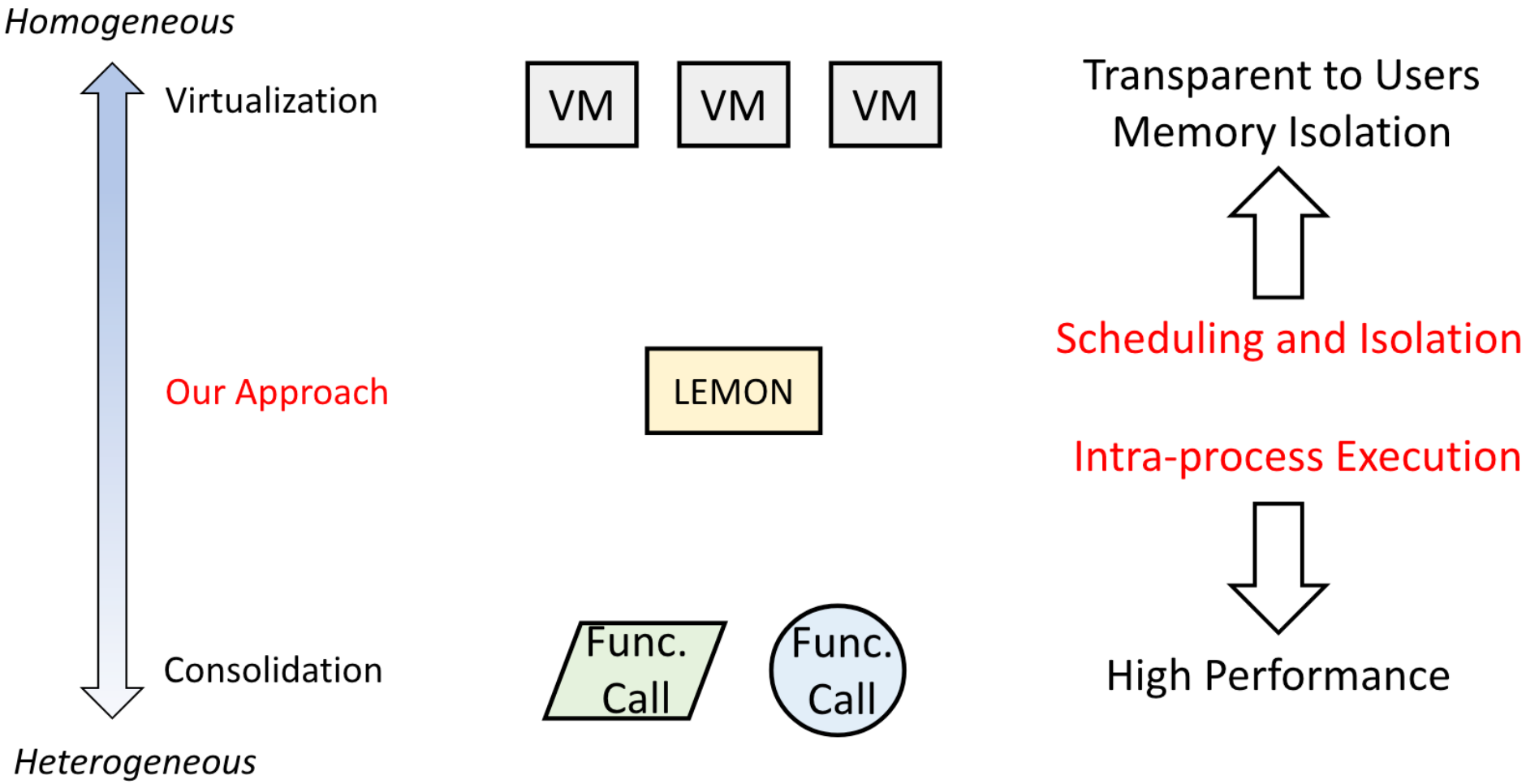
- By default, NFs process packets in an infinite loop
  - RX/TX talks directly to NIC
- Customized I/O does not modify the loop but schedules it
  - Using RX/TX as scheduling points
  - Calling TX saves context and returns to the trampolines
  - The trampolines select the next LEMON and restore its (post-TX)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

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

Evaluation

- Minimum LOC Modification to Interoperation

Heterogeneity of Real World NFs

Huge Code Base Of Real World NFs

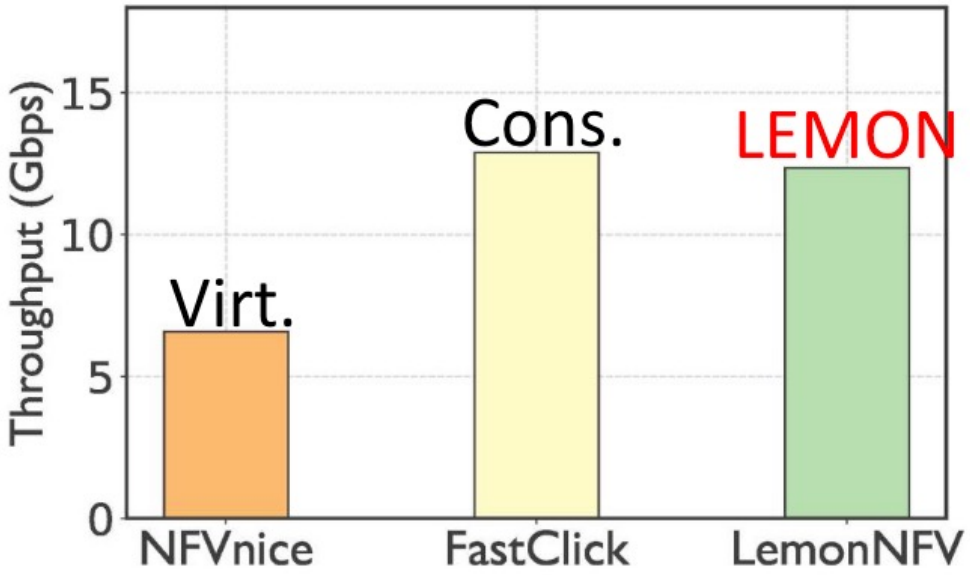
Effort of LemonNFV

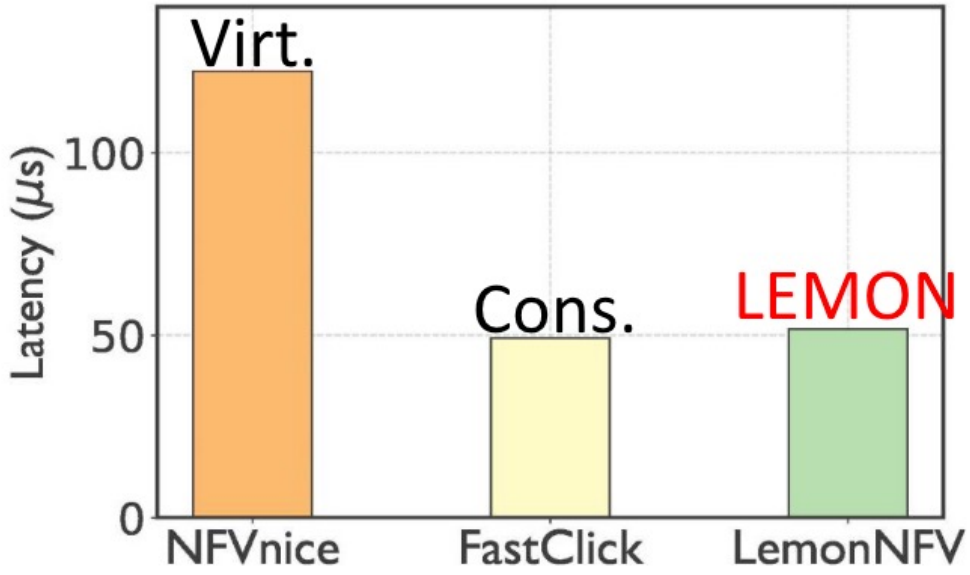
NF	Framework	Language	I/O	NF LOC	Framework LOC	Modified LOC
IDS	Rubik	C	DPDK	337	31K	2
NAT	FastClick	C++	DPDK	94	331K	2
ACL	NetBricks	Rust	DPDK	401	58K	8
CT	mOS	C	libpcap	325	139K	4
DPI	nDPI	C	libpcap	4498	121K	2

- LemonNFV consolidates heterogeneous NFs without much effort (LOC)

Evaluation

- Comparing Performance with State-Of-The-Art

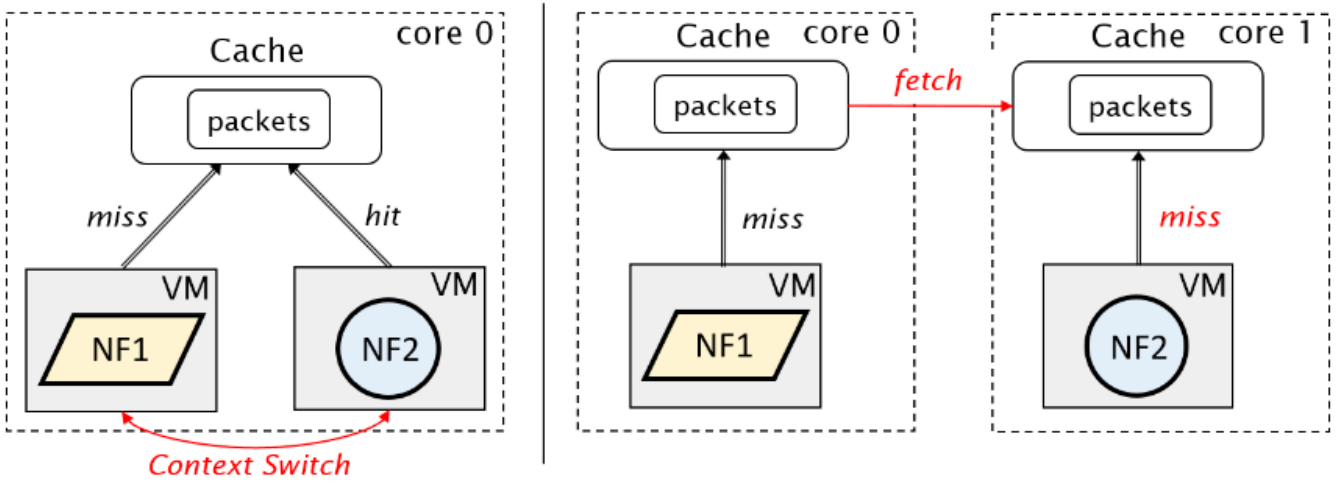


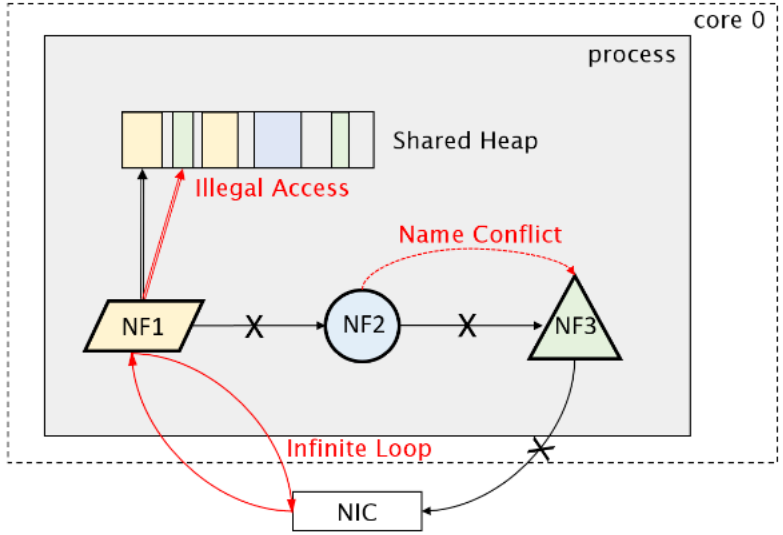


- LemonNFV consolidates heterogeneous NFs with minor overhead

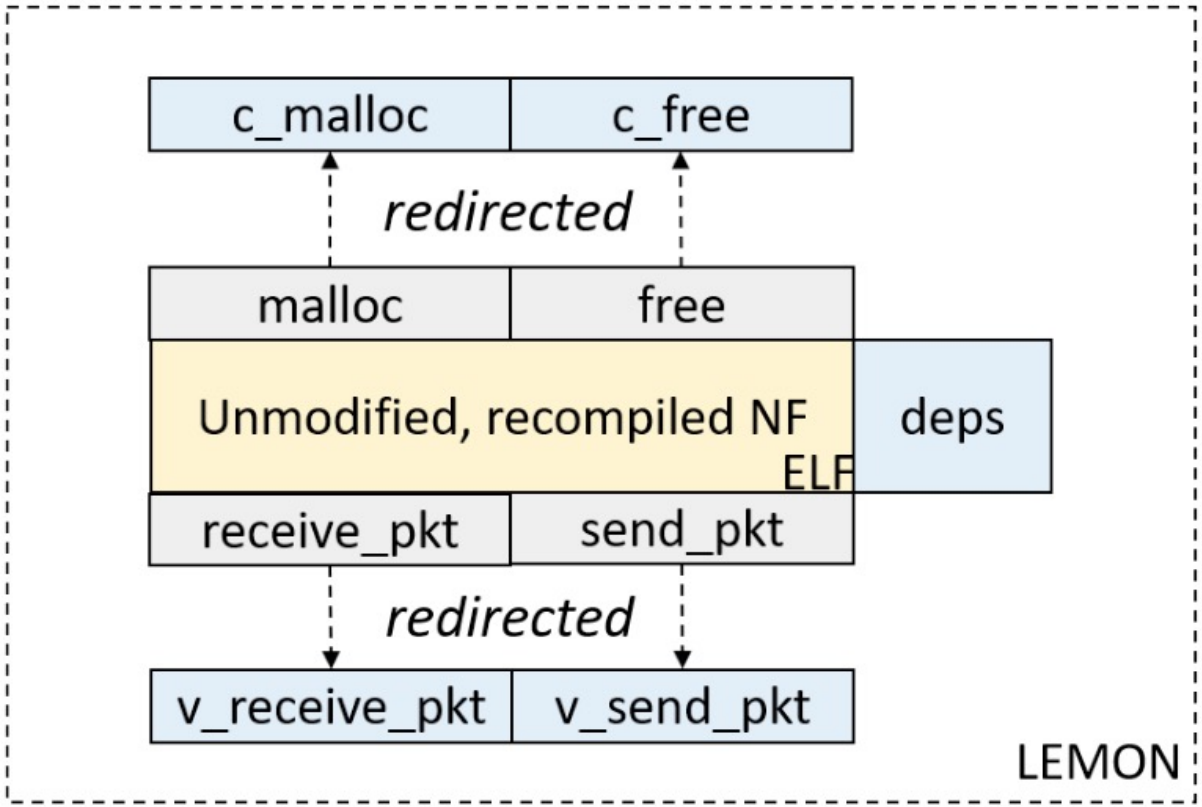
Summary

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





Summary



- LemonNFV consolidates NFs with minor overhead and effort
  - Designs a unique abstraction LEMON
  - Schedules and isolates LEMONs inside one process