We Need Kernel Interposition over the Network Dataplane

Hugo Sadok, Zhipeng Zhao, Valerie Choung, Nirav Atre, Daniel S. Berger, James C. Hoe, Aurojit Panda, Justine Sherry
Traditional Kernel Network Stack

Most popular design today (e.g., Linux, Windows, BSD)
Traditional Kernel Network Stack

Most popular design today (e.g., Linux, Windows, BSD)
Traditional Kernel Network Stack

Most popular design today (e.g., Linux, Windows, BSD)
Traditional Kernel Network Stack

Most popular design today (e.g., Linux, Windows, BSD)
Low performance due to data movement:
- Copies
- Syscalls
Kernel-Bypass Network Stack

Let applications interface directly with the NIC and implement the network stack in a library
Kernel-Bypass Network Stack

Let applications interface directly with the NIC and implement the network stack in a library.
Kernel-Bypass Network Stack

Let applications interface directly with the NIC and implement the network stack in a library.

Avoids the additional data movement imposed by the kernel stack.
Kernel bypass leads to a maintenance and manageability nightmare for administrators!
Kernel bypass leads to a maintenance and manageability nightmare for administrators!

Main Issue: Kernel bypass cannot fully replicate kernel functionality
Policy Enforcement

NIC <-> Kernel <-> Database
Policy Enforcement

NIC

Kernel

Database

Bob’s App

User
Policy Enforcement

Bob’s app is sending too much traffic, not leaving enough bandwidth for the database.
I want to prioritize traffic from the database over traffic from Bob’s app.
I want to prioritize traffic from the database over traffic from Bob’s app. I can use `tc` to set queuing disciplines!
Policy Enforcement

I want to prioritize traffic from the database over traffic from Bob’s app.

I can use `tc` to set queuing disciplines!
I want to prioritize traffic from the database over traffic from Bob’s app.

I can use tc to set queuing disciplines!
I want to prioritize traffic from the database over traffic from Bob’s app.

I can use `tc` to set queuing disciplines!
Bob set a weak password and the server got compromised with a malicious webserver.
Which application is listening on port 80?
Which application is listening on port 80?

I can use tools like `netstat/ss` to check all open sockets!
Policy Enforcement

“I want to prioritize traffic from the database over traffic from Bob’s app”

Debugging

“Which application is listening on port 80?”
Policy Enforcement

“I want to prioritize traffic from the database over traffic from Bob’s app”

Debugging

“Which application is listening on port 80?”
**Policy Enforcement**

“I want to prioritize traffic from the database over traffic from Bob’s app”

**Debugging**

“Which application is listening on port 80?”

---

**Problem: Lack of kernel interposition**
Existing Alternatives

Google Snap [SOSP ’19]
Existing Alternatives

Google Snap [SOSP ’19]

Reintroduces data movement overheads
Existing Alternatives

Google Snap [SOSP '19]
- NIC
- App 1
  - Core 0
- App 2
  - Core 1
- Snap
  - Core 2

Reintroduces data movement overheads

Microsoft AccelNet [NSDI '18]
- NIC
- vSwitch
  - App 1
- App 2

Virtual Machine
Existing Alternatives

Google Snap [SOSP ’19]

- Reintroduces data movement overheads

Microsoft AccelNet [NSDI ’18]

- Control lies with the hypervisor/no introspection into applications
So what do we *really* want?
So what do we really want?

*Logically* like the kernel network stack
So what do we *really* want?

*Logically* like the kernel network stack

*Physically* like kernel bypass
So what do we really want?

*Logically* like the kernel network stack

Physically like kernel bypass

Kernel interposition
So what do we really want?

Logically like the kernel network stack

Physically like kernel bypass

Kernel interposition

On path
So what do we really want?

Logically like the kernel network stack

Physically like kernel bypass

Kernel interposition

On path

KOPI: Kernel On-Path Interposition
So what do we *really* want?

Logically like the kernel network stack

Physically like kernel bypass

**KOPI: Kernel On-Path Interposition**

Leverage programmable SmartNICs to implement a network dataplane that is both on path and logically controlled by the kernel.
KOPI: Kernel On-Path Interposition

Leverage programmable SmartNICs to implement a network dataplane that is both on path and logically controlled by the kernel.
KOPI: Kernel On-Path Interposition

Leverage programmable SmartNICs to implement a network dataplane that is both on path and logically controlled by the kernel.
KOPI: Kernel On-Path Interposition

Leverage programmable SmartNICs to implement a network dataplane that is both on path and logically controlled by the kernel.

Can install code on the NIC to define dataplane functionality.
KOPI: Kernel On-Path Interposition

Leverage programmable SmartNICs to implement a network dataplane that is both on path and logically controlled by the kernel.

Can install code on the NIC to define dataplane functionality.

Number of commits on Linux last year:
- net/netfilter: 377
- net/sched: 249
KOPI: Kernel On-Path Interposition

Leverage programmable SmartNICs to implement a network dataplane that is both on path and logically controlled by the kernel.

Can install code on the NIC to define dataplane functionality.

Number of commits on Linux last year
net/netfilter: 377
net/sched: 249
KOPI: Kernel On-Path Interposition

Leverage programmable SmartNICs to implement a network dataplane that is both on path and logically controlled by the kernel.

Can install code on the NIC to define dataplane functionality.

Control plane operations.

Number of commits on Linux last year:
- net/netfilter: 377
- net/sched: 249
KOPI: Kernel On-Path Interposition

Leverage programmable SmartNICs to implement a network dataplane that is both on path and logically controlled by the kernel.

Can install code on the NIC to define dataplane functionality.

Number of commits on Linux last year:
- net/netfilter: 377
- net/sched: 249
Overview of Norman: An OS that Implements KOPI
Overview of Norman: An OS that Implements KOPI

Norman OS

- KOPI Dataplane
- Kernel (Control Plane)
- Library
- App

SmartNIC

CPU
Overview of Norman: An OS that Implements KOPI

Uses an FPGA SmartNIC to implement the dataplane.
Overview of Norman: An OS that Implements KOPI

Norman OS

- Uses an FPGA SmartNIC to implement the dataplane.
- Responsible for all the interposition logic: filters, queuing disciplines, packet sniffing, etc.

KOPI Dataplane

Kernel (Control Plane)

Library

App

SmartNIC

CPU
Overview of Norman: An OS that Implements KOPI

Norman OS

- Uses an FPGA SmartNIC to implement the dataplane.
- Responsible for all the interposition logic: filters, queuing disciplines, packet sniffing, etc.
- Runs dataplane logic that does not require interposition.
Overview of Norman: An OS that Implements KOPI

- Norman OS
  - Uses an FPGA SmartNIC to implement the dataplane.
  - Responsible for all the interposition logic: filters, queuing disciplines, packet sniffing, etc.
  - Runs dataplane logic that does not require interposition.

- Based on Linux. Responsible for all control plane operations.
Overview of Norman: An OS that Implements KOPI

Uses an FPGA SmartNIC to implement the dataplane.

Based on Linux. Responsible for all control plane operations.

Responsible for all the interposition logic: filters, queuing disciplines, packet sniffing, etc.

Runs dataplane logic that does not require interposition.
Overview of Norman: An OS that Implements KOPI

- Uses an FPGA SmartNIC to implement the dataplane.
- Responsible for all the interposition logic: filters, queuing disciplines, packet sniffing, etc.
- Runs dataplane logic that does not require interposition.
- Based on Linux. Responsible for all control plane operations.

Norman OS

Kernel (Control Plane)

Library

App

SmartNIC

CPU
Overview of Norman: An OS that Implements KOPI

Uses an FPGA SmartNIC to implement the dataplane.

Responsible for all the interposition logic: filters, queuing disciplines, packet sniffing, etc.

Runs dataplane logic that does not require interposition.

Based on Linux. Responsible for all control plane operations.

- iptables
- tc
Overview of Norman: An OS that Implements KOPI

- Uses an FPGA SmartNIC to implement the dataplane.
- Responsible for all the interposition logic: filters, queuing disciplines, packet sniffing, etc.
- Runs dataplane logic that does not require interposition.
- Based on Linux. Responsible for all control plane operations.
Overview of Norman: An OS that Implements KOPI

**Norman OS**

- Uses an FPGA SmartNIC to implement the dataplane.
- Responsible for all the interposition logic: filters, queuing disciplines, packet sniffing, etc.
- Runs dataplane logic that does not require interposition.
- Based on Linux. Responsible for all control plane operations.

**Dataplane**

- KOPI
- Filter rule

**Kernel (Control Plane)**

- iptables
- tc

**Library**

- connect
- write

**CPU**

- App
Open Challenges
Open Challenges

① Can Norman design scale to support enough connections?
Open Challenges

① Can Norman design scale to support enough connections?

② How to make an FPGA reconfigurable enough for our purposes?
Conclusion
Conclusion

Kernel interposition is essential
Conclusion

Kernel interposition is **essential**

The **KOPI architecture** gives a path to restore interposition without reintroducing overheads