Fine-Grained Fault Tolerance using Device Checkpoints

Asim Kadav
with Matthew Renzelmann and Michael M. Swift
University of Wisconsin-Madison

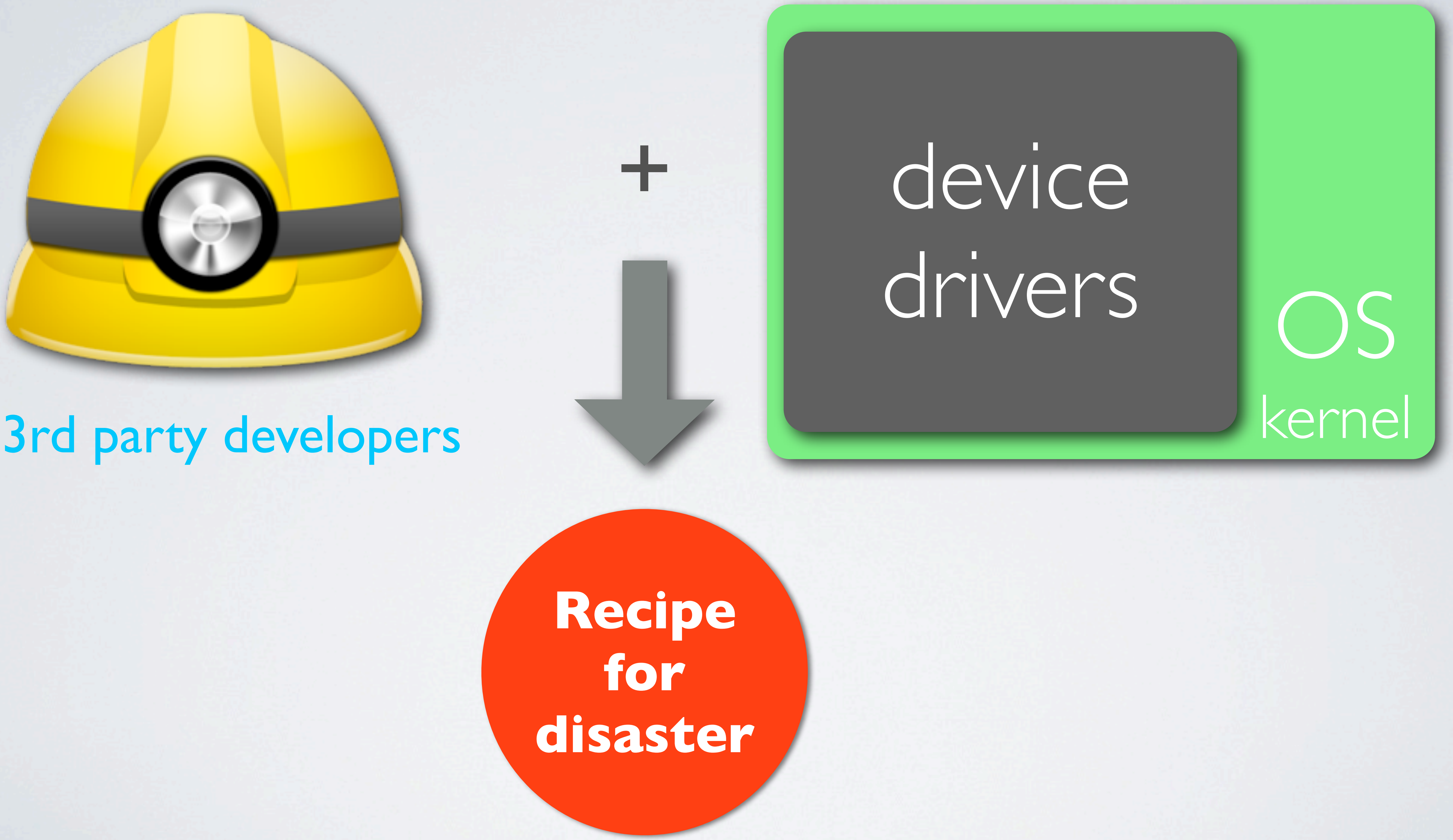
The (old) elephant in the room

3rd party developers

+

The (old) elephant in the room

The (old) elephant in the room



Past work mostly looks at detection and isolation

Improvement	System	Validation		
		Drivers	Bus	Classes
New functionality	Shadow driver migration [OSR09]	1	1	1
	RevNIC [Eurosys 10]	1	1	1
Reliability	Nooks [SOSP 03]	6	1	2
	XFI [OSDI 06]	2	1	1
	CuriOS [OSDI 08]	2	1	2
Type Safety	SafeDrive [OSDI 06]	6	2	3
	Singularity [Eurosys 06]	1	1	1
Specification	Nexus [OSDI 08]	2	1	2
	Termite [SOSP 09]	2	1	2
Static analysis tools	Windows SDV [Eurosys 06]	All	All	All
	Coverity [CACM 10]	All	All	All
	Cocinelle [Eurosys 08]	All	All	All

Past work mostly looks at detection and isolation

Improvement	System	Validation		
		Drivers	Bus	Classes
New functionality	Shadow driver migration [OSR09]	1	1	1
	RevNIC [Eurosys 10]	1	1	1
Reliability	Nooks [SOSP 03]	6	1	2
	XFI [OSDI 06]	2	1	1
	CuriOS [OSDI 08]	2	1	2
Type Safety	SafeDrive [OSDI 06]	6	2	3
	Singularity [Eurosys 06]	1	1	1
Specification	Nexus [OSDI 08]	2	1	2
	Termite [SOSP 09]	2	1	2

Large kernel subsystems and validity of few device types result in limited adoption of research solutions

Past work mostly looks at detection and isolation

Improvement	System	Validation		
		Drivers	Bus	Classes
New functionality	Shadow driver migration [OSR09]	1	1	1
	RevNIC [Eurosys 10]	1	1	1
Reliability	Nooks [SOSP 03]	6	1	2
	XFI [OSDI 06]	2	1	1
	Singularity [Eurosys 06]	1	1	1
Specification	Nexus [OSDI 08]	2	1	2
	Termite [SOSP 09]	2	1	2
Static analysis tools	Windows SDV [Eurosys 06]	All	All	All
	Coverity [CACM 10]	All	All	All
	Cocinelle [Eurosys 08]	All	All	All

Limited kernel changes + Applicable to lots of drivers => Real Impact

Past work mostly looks at detection and isolation

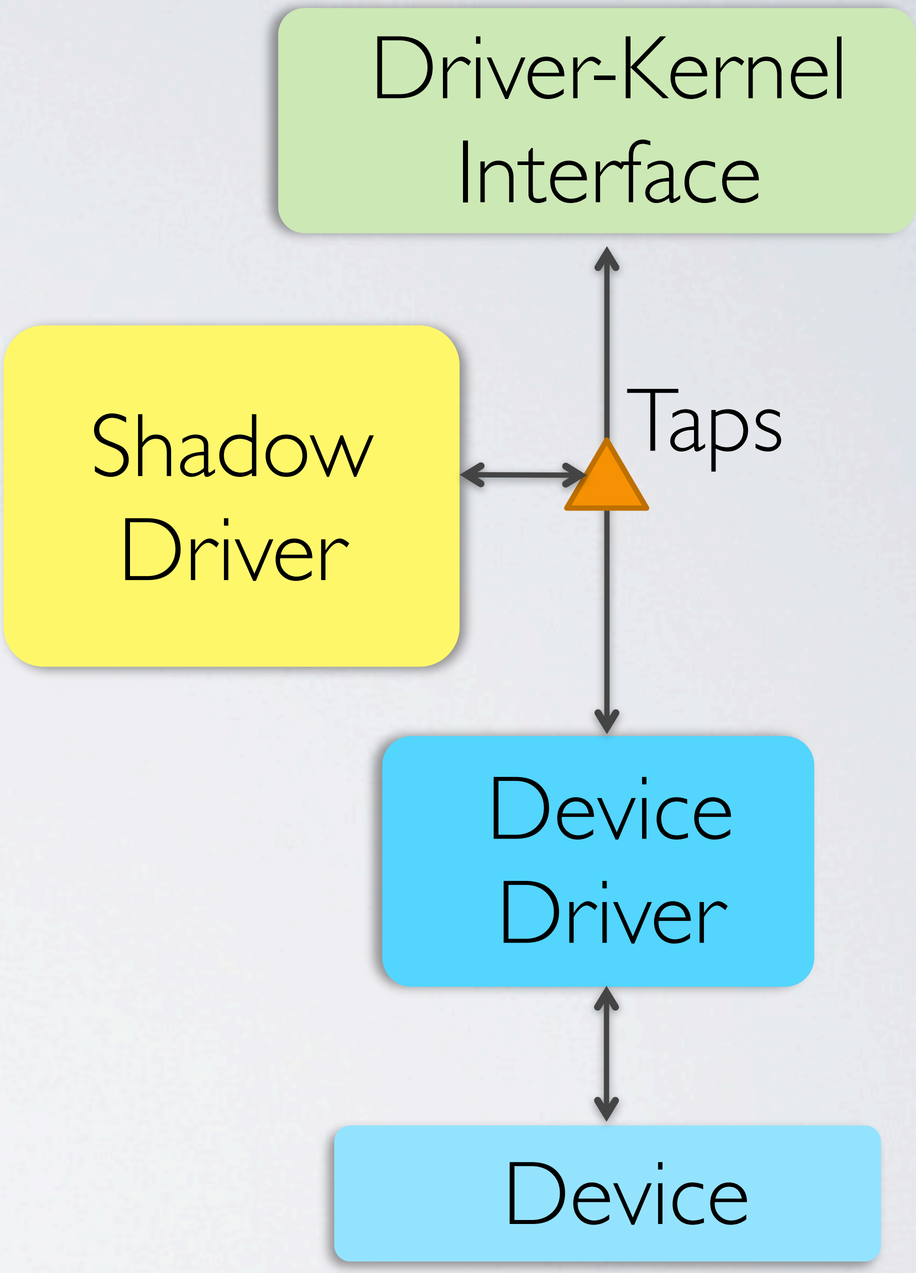
Improvement	System	Validation		
		Drivers	Bus	Classes
New functionality	Shadow driver migration [OSR09]	1	1	1
	RevNIC [Eurosys 10]	1	1	1
Reliability	Nooks [SOSP 03]	6	1	2
	XFI [OSDI 06]	2	1	1
	Singularity	1	1	1
Specification	Nexus [OSDI 08]	2	1	2
	Termite [SOSP 09]	2	1	2

Limited kernel changes + Applicable to lots of drivers => Real Impact

Goal: Improve recovery with complete solutions that can be applied to many drivers

State of the art in recovery: Shadow drivers

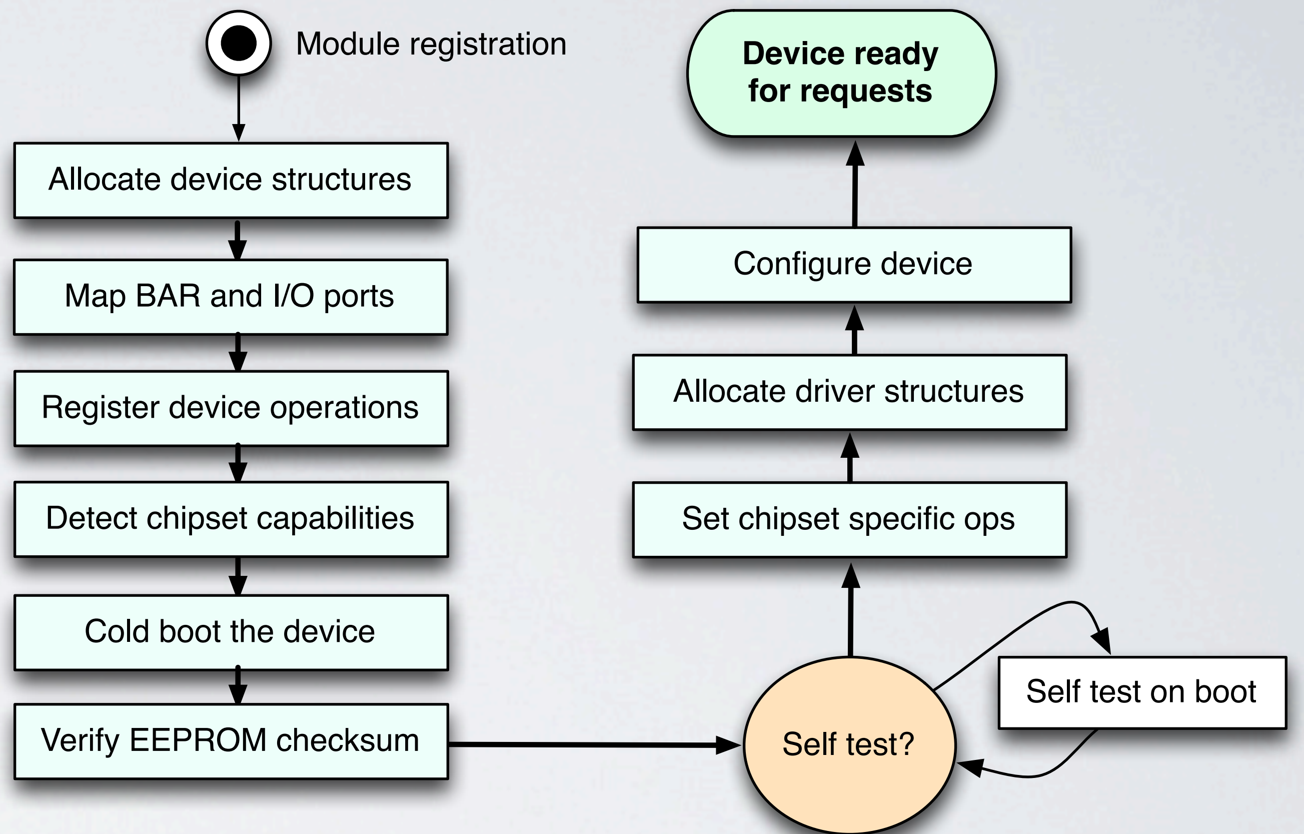
- **Carburizer calls generic recovery service if check fails**
- **Low cost transparent recovery**
  - ★ **Based on shadow drivers**
  - ★ **Records state of driver at all times**
  - ★ **Transparently restarts and replays recorded state on failure**



Swift [OSDI '04]

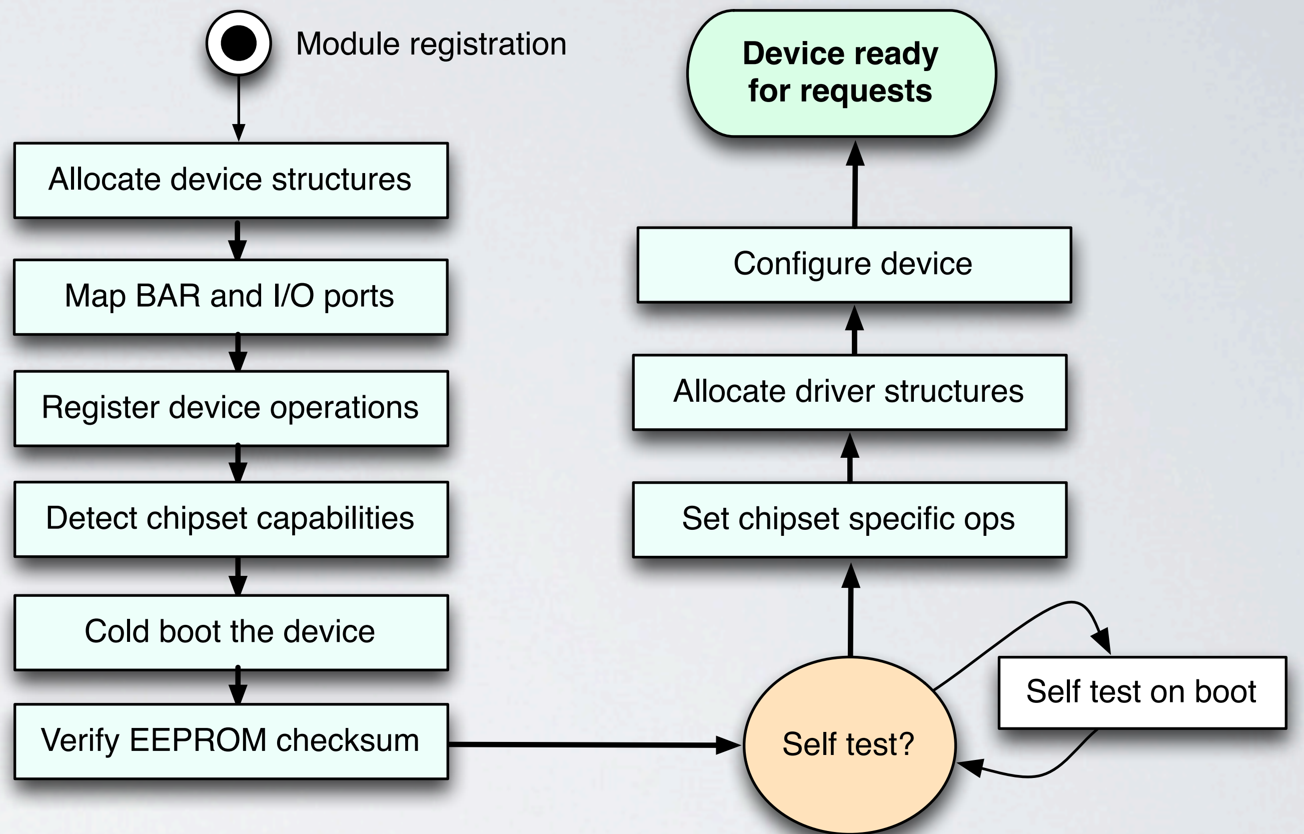
Recovery Performance: Device initialization is slow

- **Multi-second device probe**
  - **Identify device**
  - **Cold boot device**
  - **Setup device/driver structures**
  - **Configuration/Self-test**



Recovery Performance: Device initialization is slow

- **Multi-second device probe**
  - **Identify device**
  - **Cold boot device**
  - **Setup device/driver structures**
  - **Configuration/Self-test**

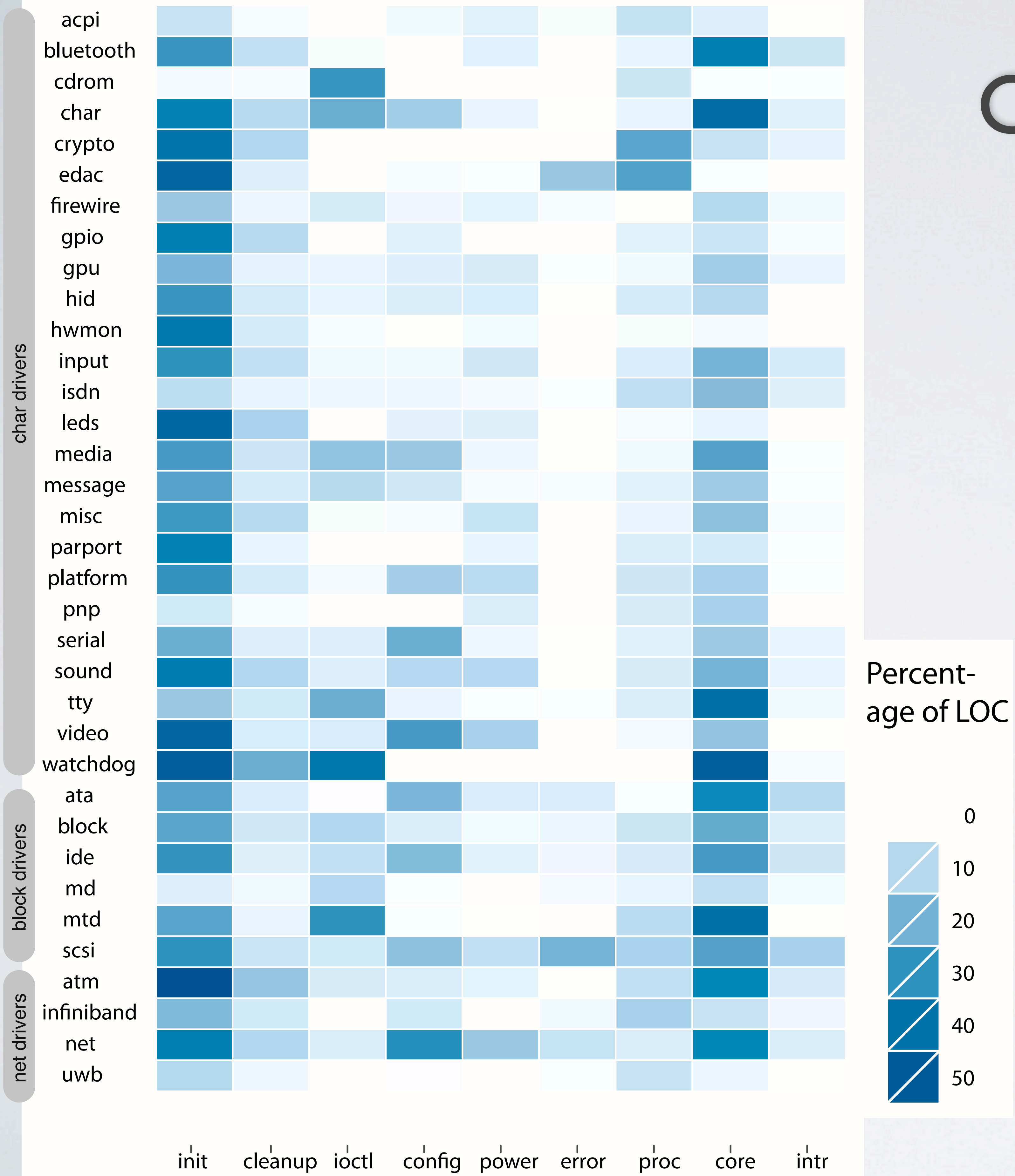


- **What does it hurt?**
  - **Fault tolerance: Driver recovery**
  - **Virtualization: Live migration, cloning, consolidation**
  - **OS functions: Boot, upgrade, NVM checkpoints**

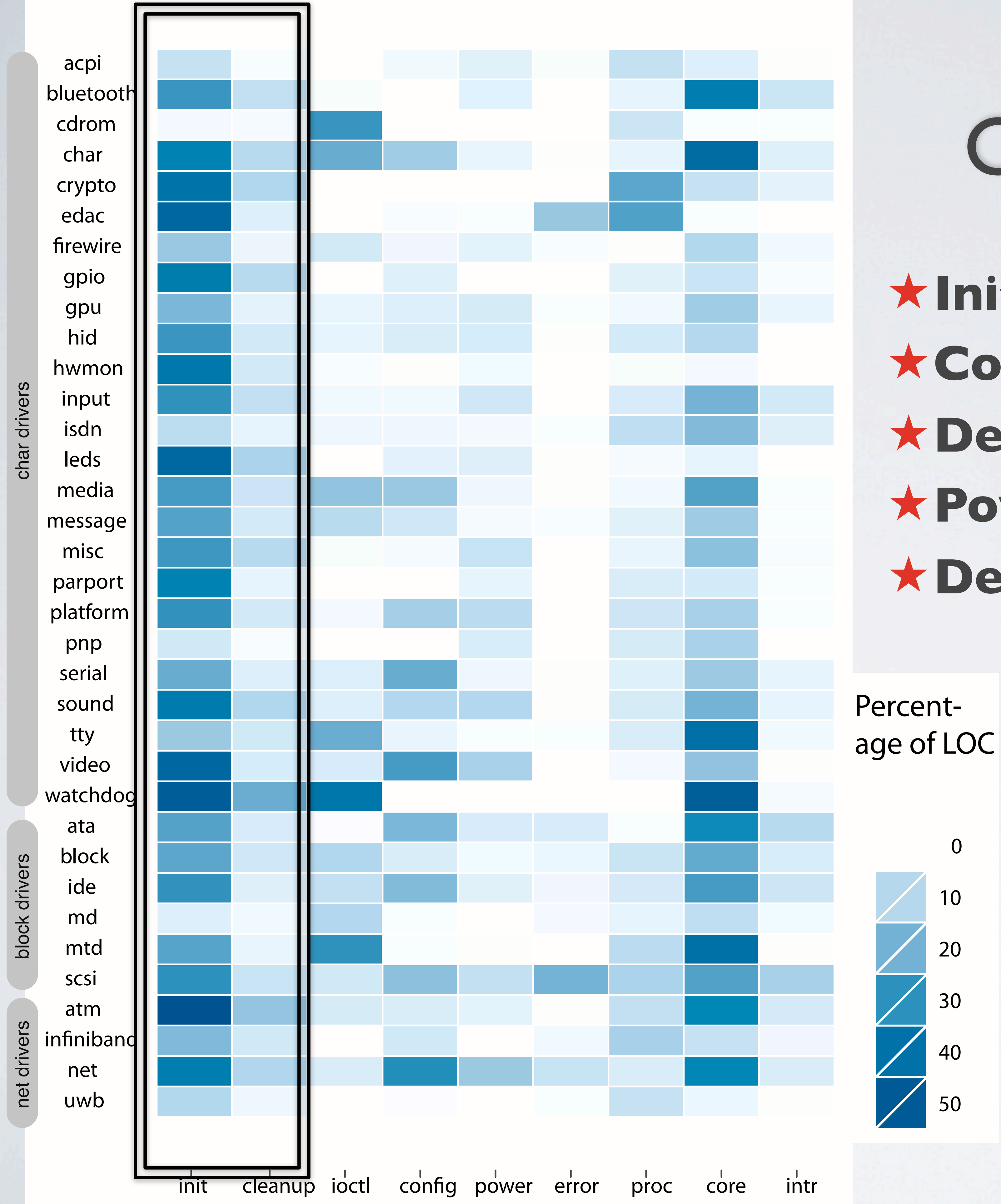
Driver Code Characteristics

★ **"Understanding Modern Device Drivers"** ASPLOS 2012

Driver Code Characteristics



★ **"Understanding Modern Device Drivers"** ASPLOS 2012

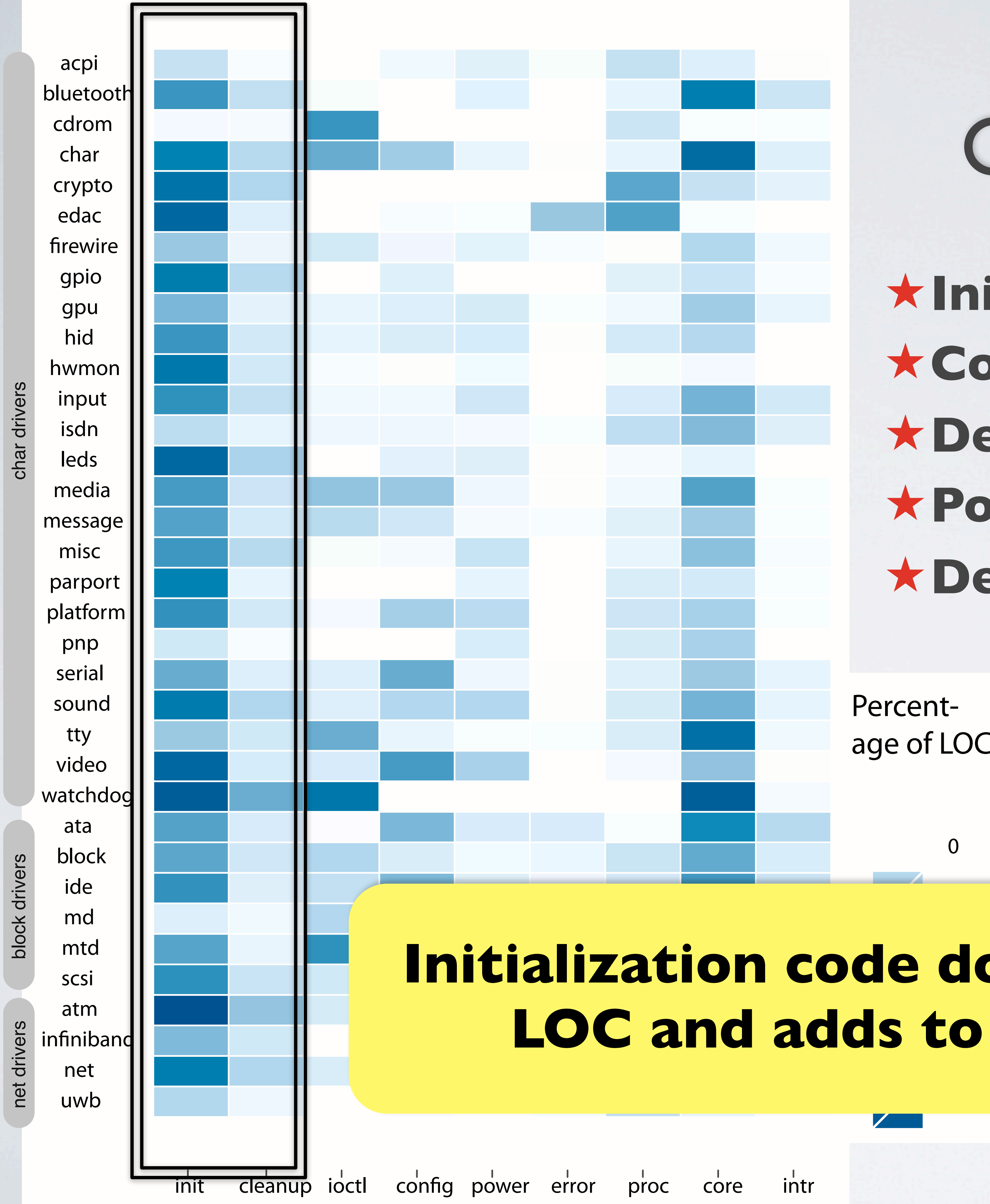


Driver Code Characteristics

- ★ **Initialization/cleanup – 36%**
- ★ **Core I/O & interrupts – 23%**
- ★ **Device configuration – 15%**
- ★ **Power management – 7.4%**
- ★ **Device ioctl – 6.2%**

★ **"Understanding Modern Device Drivers"** **ASPLOS 2012**

Driver Code Characteristics

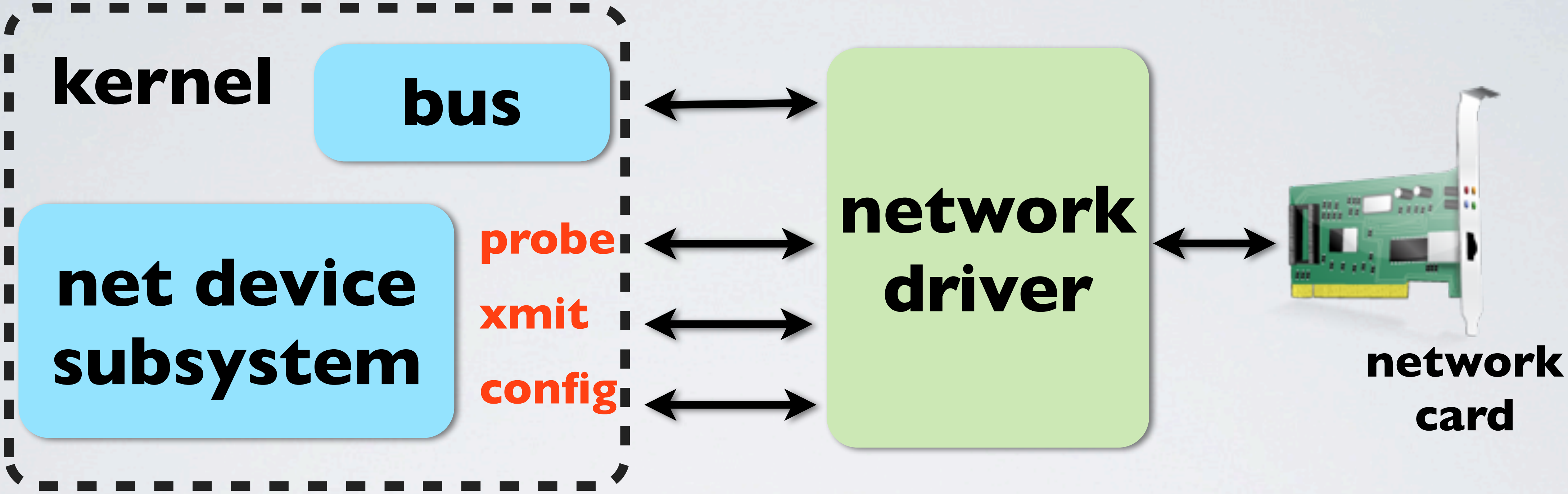


- ★ **Initialization/cleanup – 36%**
- ★ **Core I/O & interrupts – 23%**
- ★ **Device configuration – 15%**
- ★ **Power management – 7.4%**
- ★ **Device ioctl – 6.2%**

Initialization code dominates driver LOC and adds to complexity

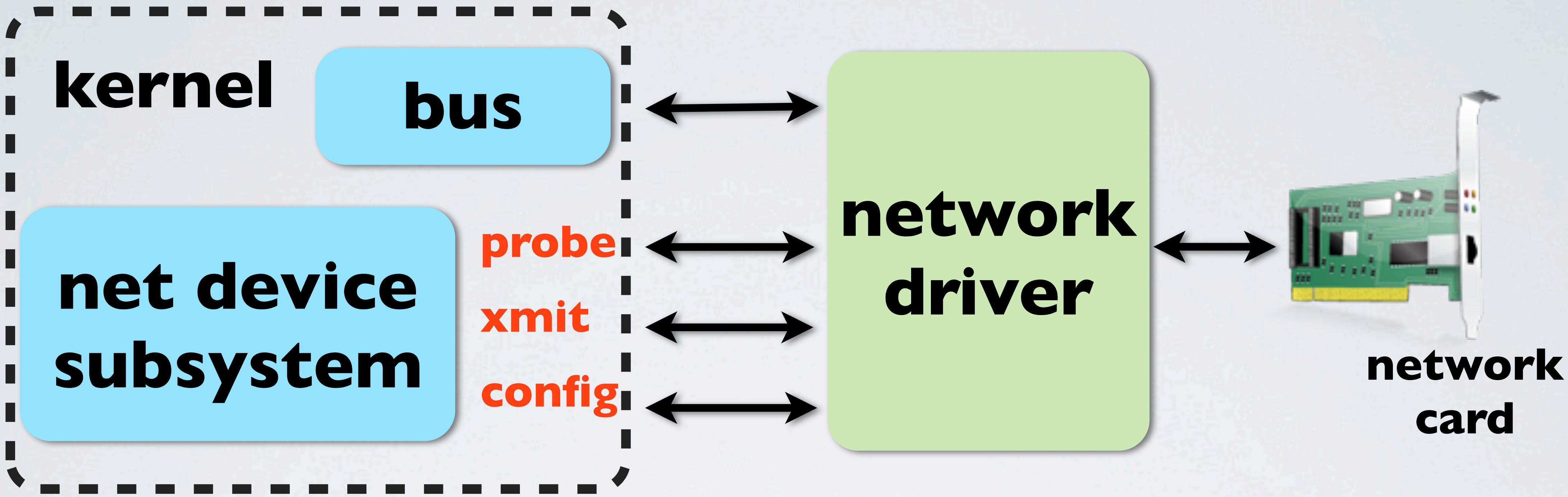
★ **"Understanding Modern Device Drivers"** **ASPLOS 2012**

Recovery works by interposing class defined entry points



- Class definition includes:
  - Callbacks registered with the bus, device and kernel subsystem

Recovery works by interposing class defined entry points



- Class definition includes:
  - Callbacks registered with the bus, device and kernel subsystem

How many drivers follow class behavior?

Restart/replay doesn't work with all drivers

- **Non-class behavior stems from:**
  - **Load time parameters, procfs and sysfs interactions, unique ioctls**

```
... qlcnic_sysfs_write_esw_config (...)       {
  ...
      switch (esw_cfg[i].op_mode) {
      case QLCNIC_PORT_DEFAULTS:
           qlcnic_set_eswitch_...(...,&esw_cfg[i]);
          ...
      case QLCNIC_ADD_VLAN:
          qlcnic_set_vlan_config(...,&esw_cfg[i]);
          ...
      case QLCNIC_DEL_VLAN:
          esw_cfg[i].vlan_id = 0;
          qlcnic_set_vlan_config(...,&esw_cfg[i]);
          ...
```

Drivers/net/qlcnic/qlcnic_main.c: Qlogic driver(network class)

★ **"Understanding Modern Device Drivers"** ASPLOS 2012

Restart/replay doesn't work with all drivers

- **Non-class behavior stems from:**
  - **Load time parameters, procfs and sysfs interactions, unique ioctls**

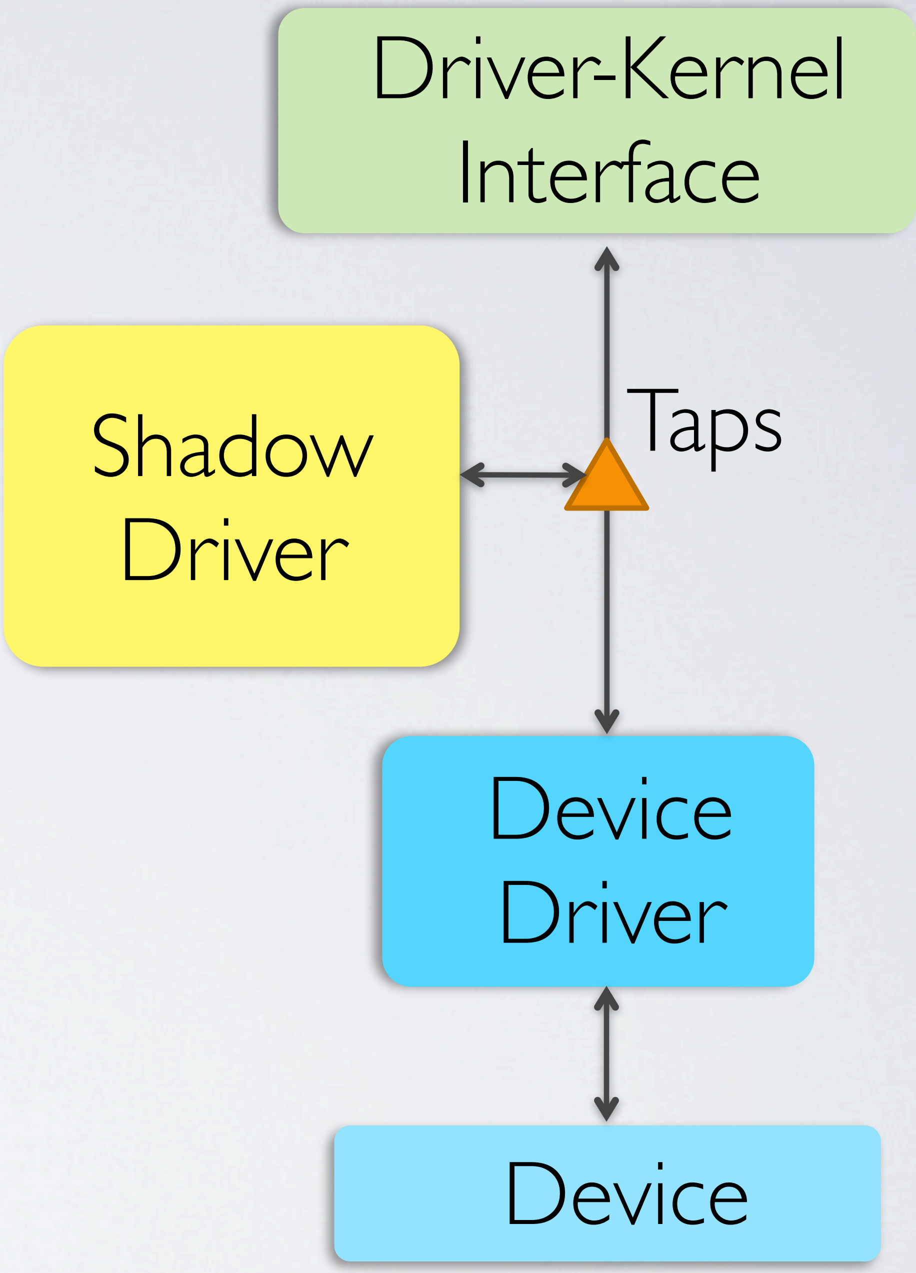
- **Results as measured by our analyses:**
  - **36% of drivers use load time parameters**
  - **16% of drivers use proc /sysfs support**

- **Overall, 44% of drivers do not conform to class behavior and recovery will not work correctly for these drivers**

- **"Understanding Modern Device Drivers"** **ASPLOS 2012**

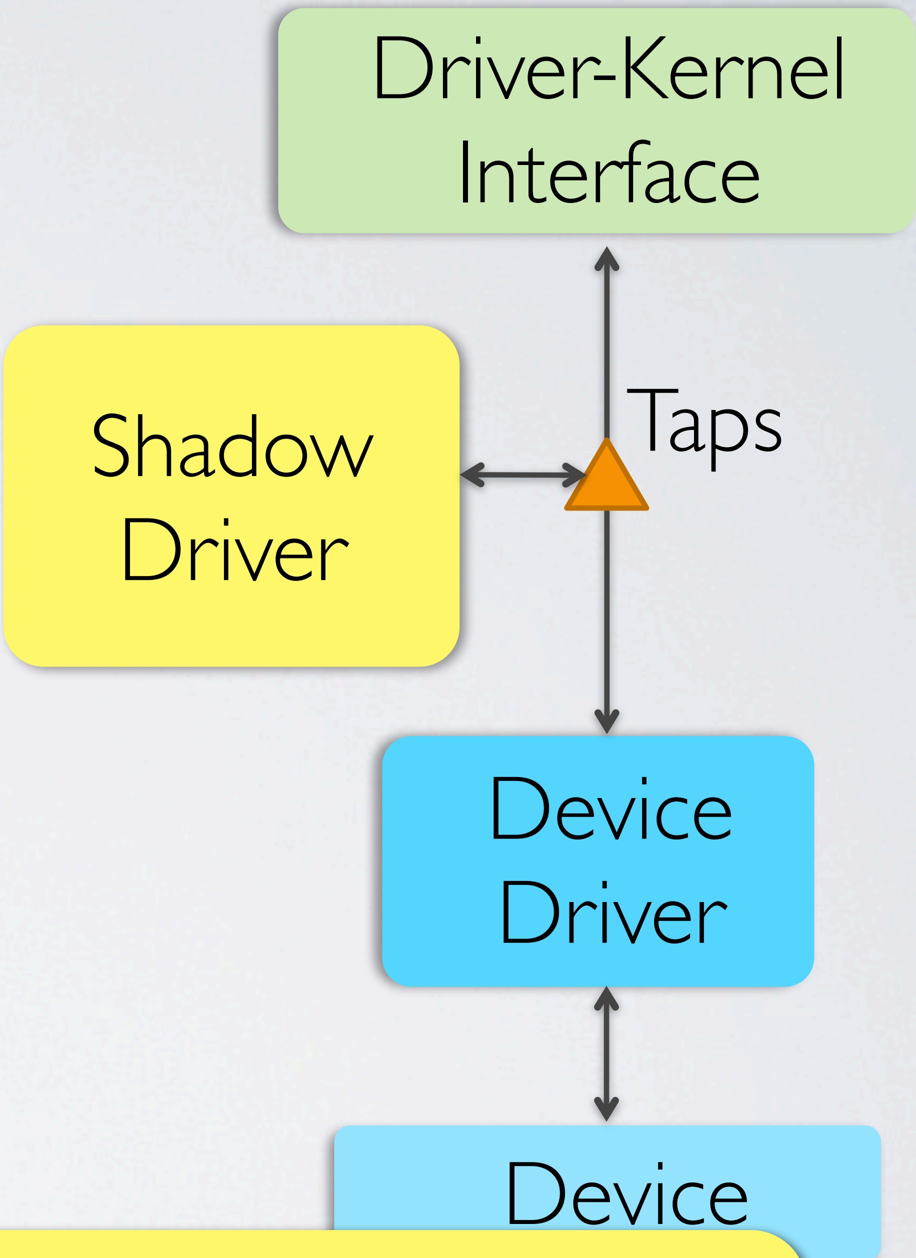
Limitations of restart/replay recovery

- **Device save/restore limited to restart/replay**
  - **Slow: Device initialization is complex (multiple seconds)**
  - **Incomplete: Unique device semantics not captured**
  - **Hard: Need to be written for every class of drivers**
  - **Large changes: Introduces new large kernel subsystems**



Limitations of restart/replay recovery

- **Device save/restore limited to restart/replay**
  - **Slow: Device initialization is complex (multiple seconds)**
  - **Incomplete: Unique device semantics not captured**
  - **Hard: Need to be written for every class of drivers**
  - **Large changes: Introduces new large kernel subsystems**



Checkpoint/restore of device and driver state removes the need to reboot device and replay state

Fine-Grained Fault Tolerance (FGFT)

Goal: Fault isolation and recovery system based on "pay as you go" failure model

Fine-Grained Isolation

- **Ability to run select entry points as transactions**

Checkpoint based recovery

- **Provides fast and correct recovery semantics**

- **Requires incremental changes to drivers and has low overhead**

Outline

Introduction

Fine-grained isolation

Checkpoint based recovery

Conclusion

Outline

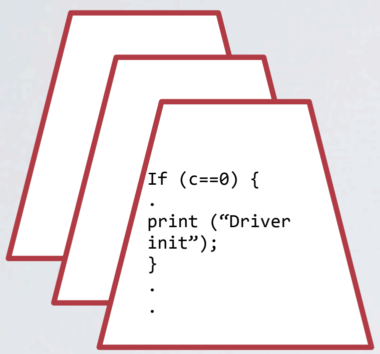
Introduction

Fine-grained isolation

Checkpoint based recovery

Conclusion

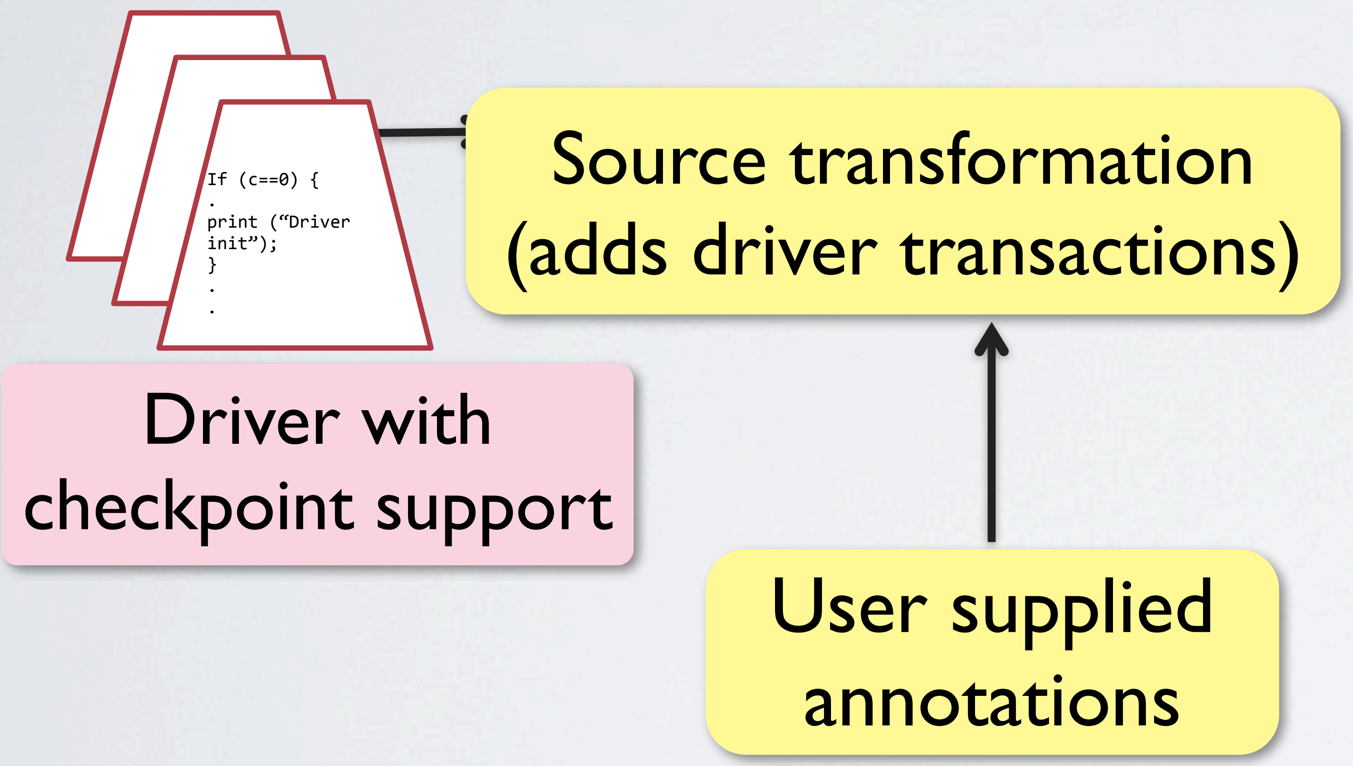
FGFT overview



Driver with checkpoint support

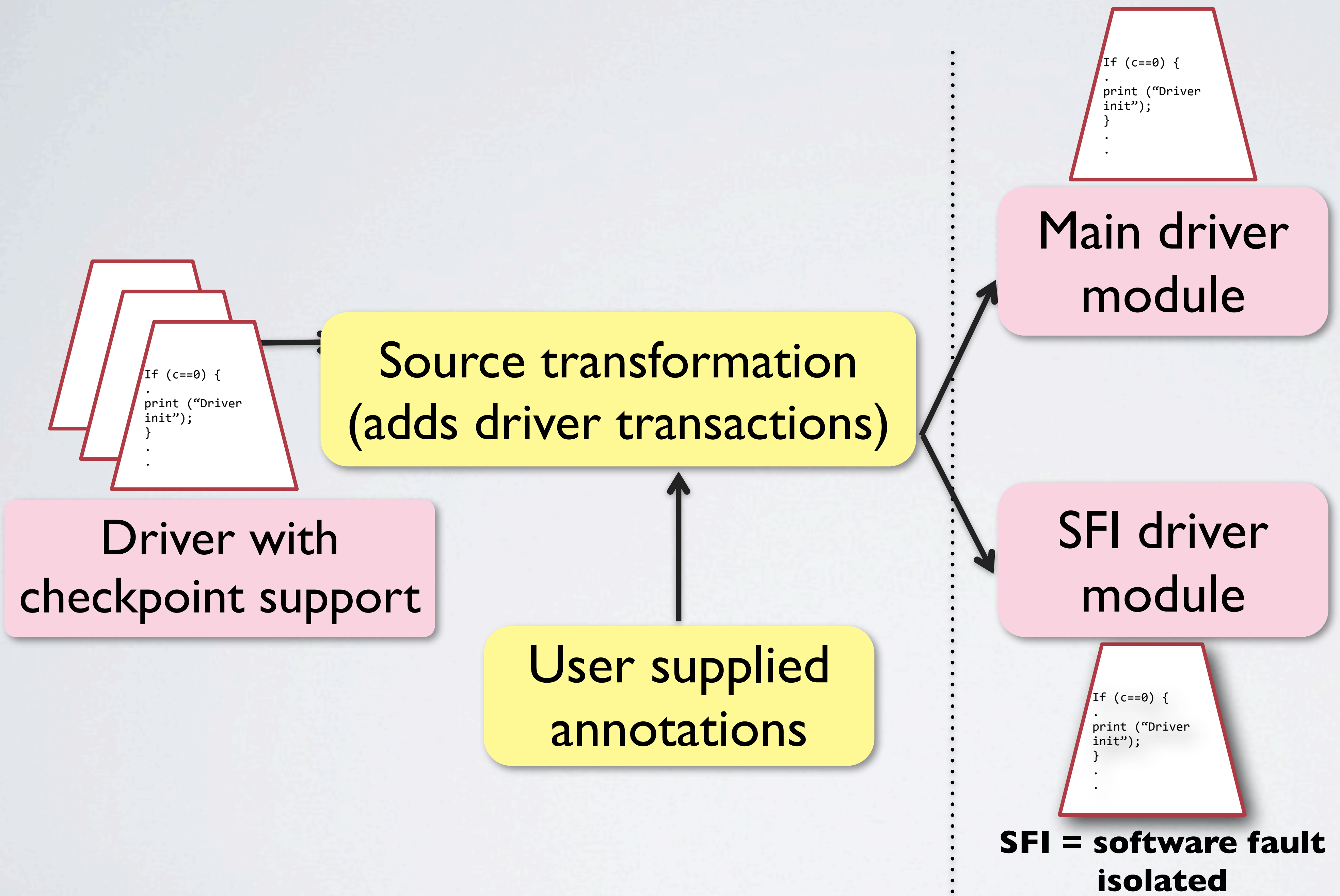
Static modifications

FGFT overview



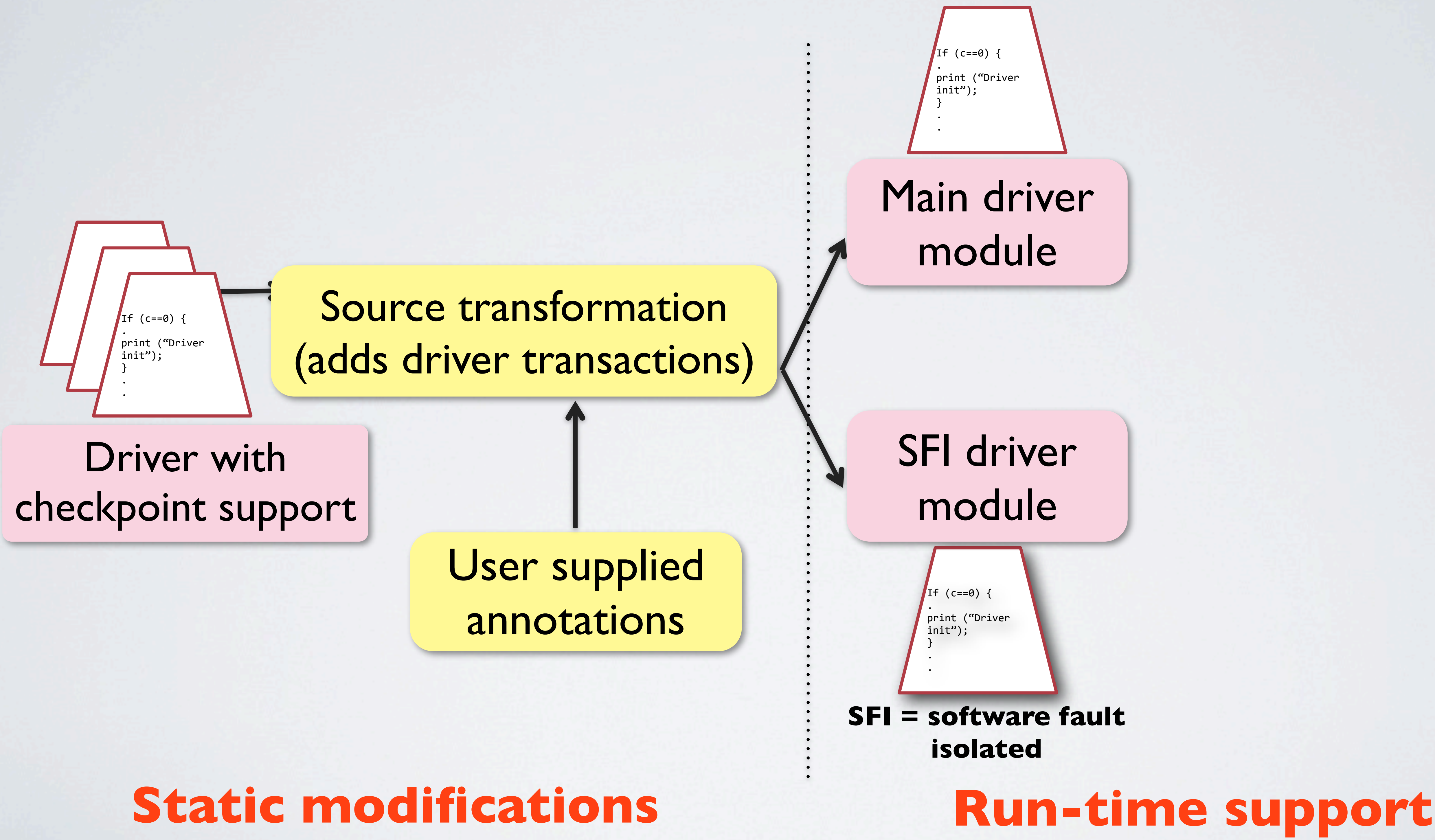
Static modifications

FGFT overview

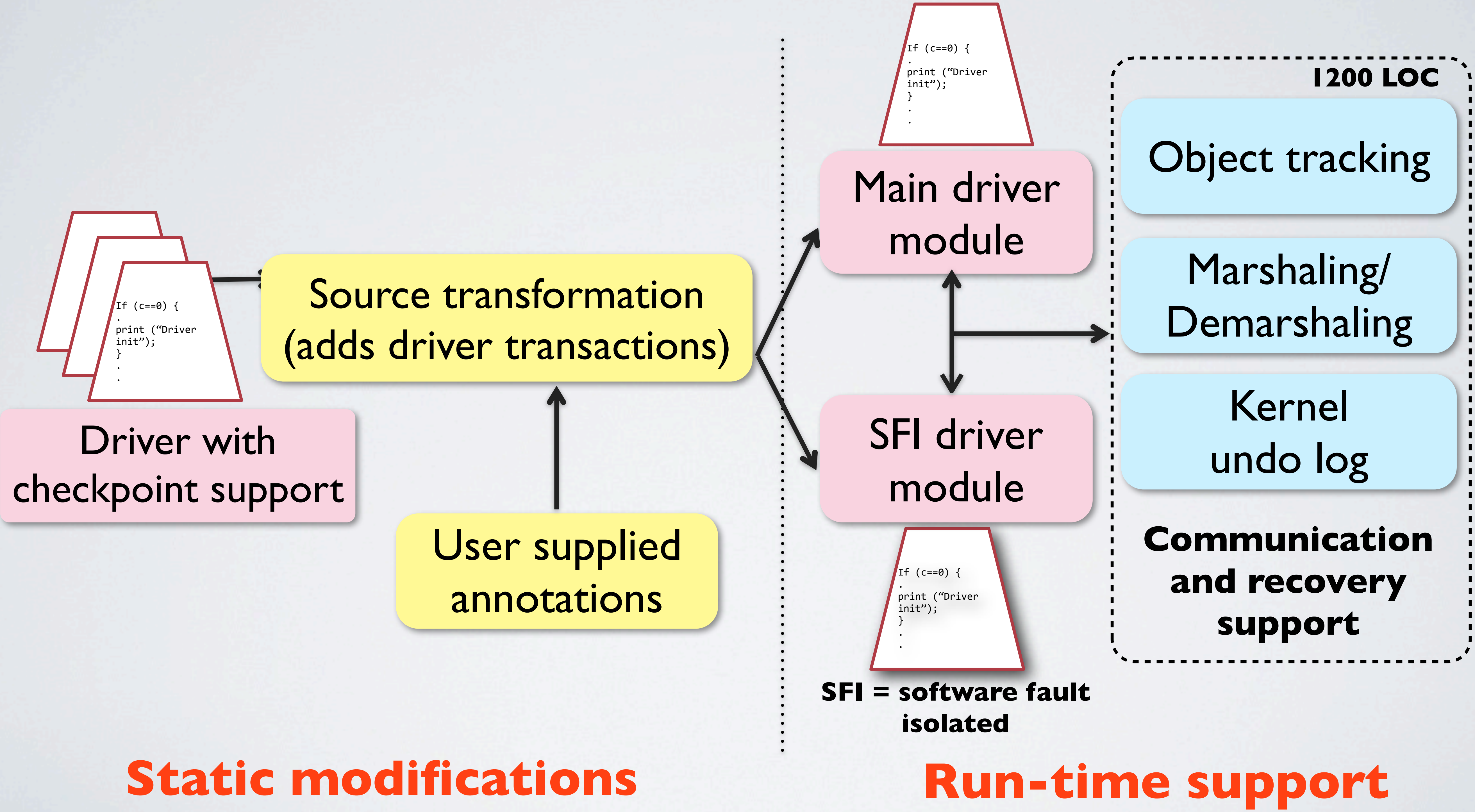


Static modifications

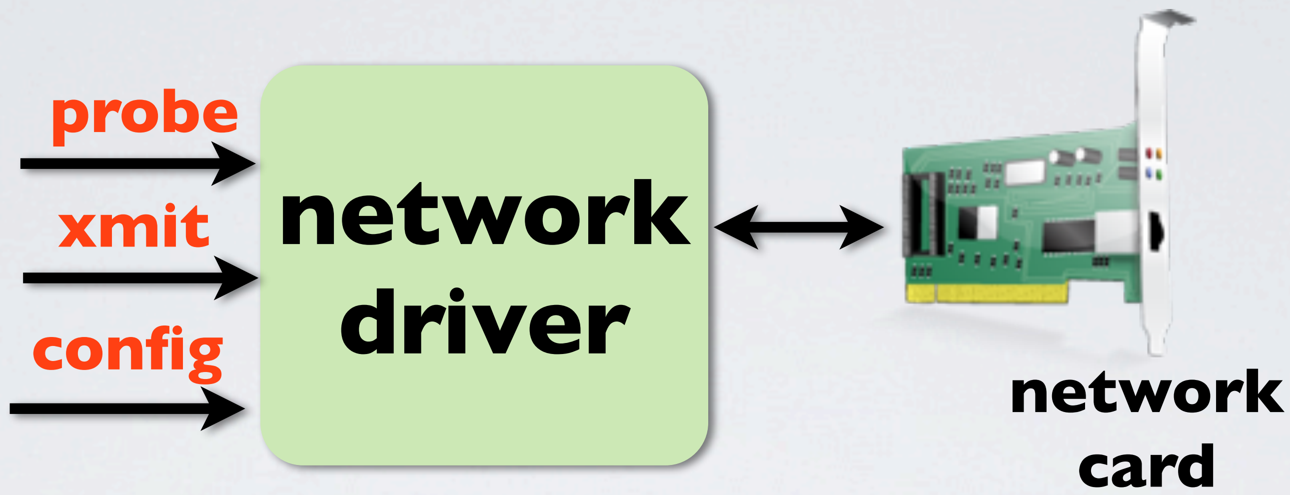
FGFT overview



FGFT overview

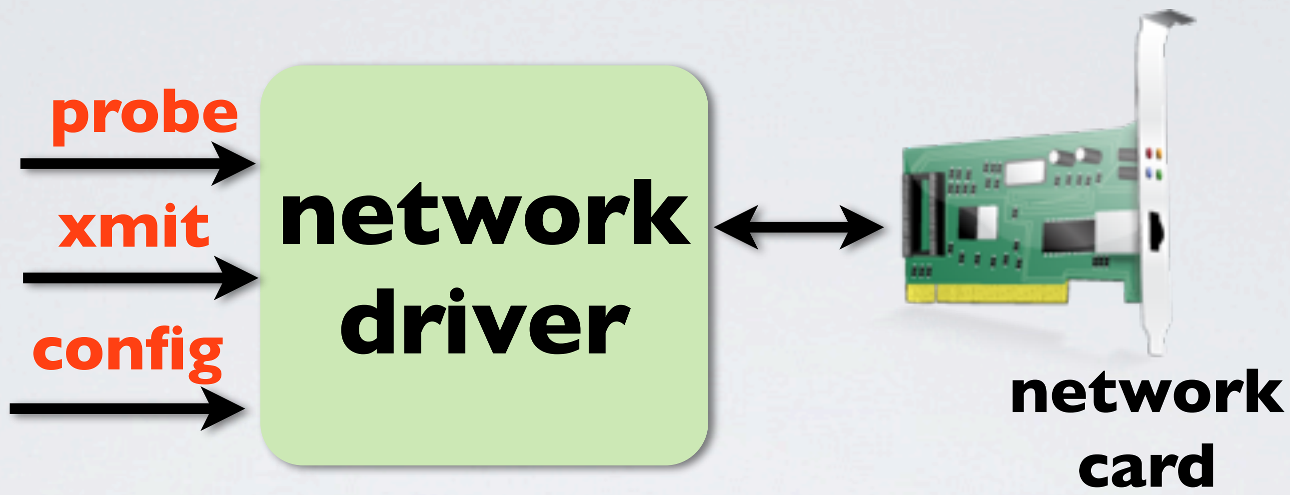


Fault model in FGFT



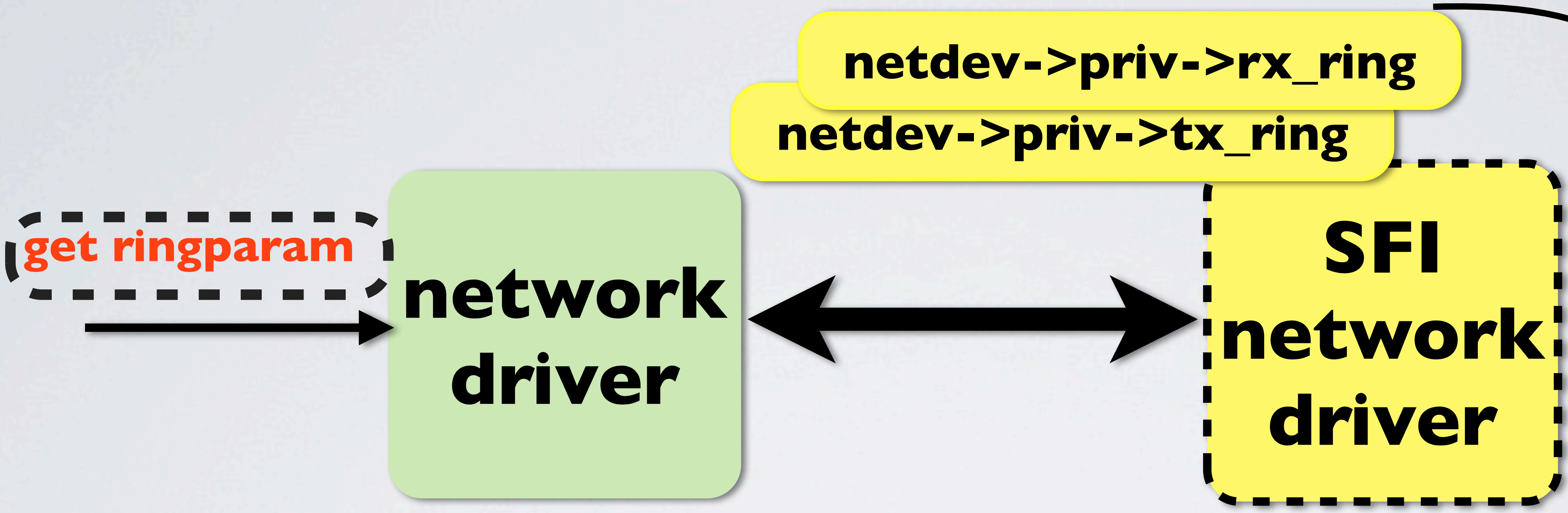
- Can be applied to untested code, statically and dynamically detected suspicious entry points
- Detect and recover from:
  - Memory errors like NULL pointer accesses
  - Structural errors like malformed structures
  - Processor exceptions like divide by zero, stack corruption

Fault model in FGFT



- ★ **Provide fault tolerance to specific driver entry points**
- ★ **Can be applied to untested code, statically and dynamically detected suspicious entry points**
- ★ **Detect and recover from:**
  - ★ **Memory errors like NULL pointer accesses**
  - ★ **Structural errors like malformed structures**
  - ★ **Processor exceptions like divide by zero, stack corruption**

Transactional support through code generation



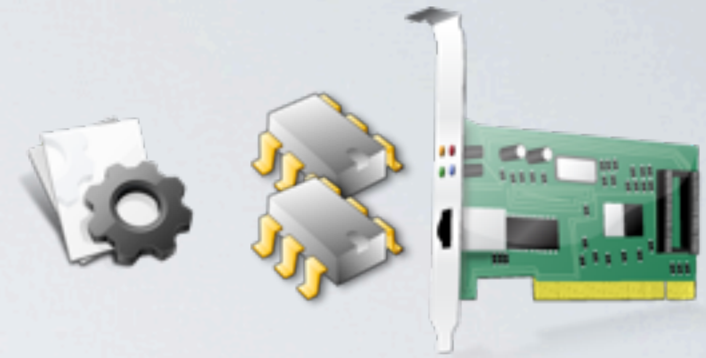
Range Table

Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

- ★ **Generate code to run driver invocations on a separate stack with a copy of parameters**
- ★ **Reduce copy overhead by copying only referenced fields in driver and kernel structures to a range table**
- ★ **Instrument all memory references in SFI module to compare accesses against copied fields in range table**

Resource access during isolated execution

- Device registers and I/O memory
  - Grant drivers full access to devices
  - Restore device checkpoint in case of failure



- Locks: Spinlocks and semaphores
  - Grants read access to locks
  - Maintain kernel log of locks acquired
  - Release locks at the end of entry point/failures

- Kernel resources like memory
  - All allocations generate range table entry
  - Maintain kernel log of all acquired resources
  - Free resources on failures

`malloc ()` →

Range Table

Address	Access rights
0xffffa000	Read
0xffffa008	Write
0xffffa00a	Read

Outline

Introduction

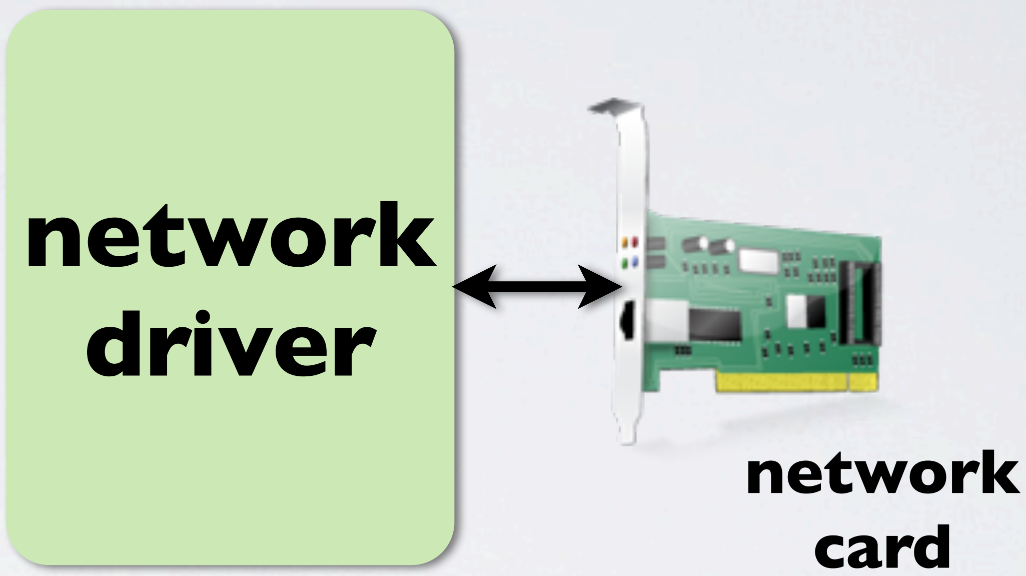
Fine-grained isolation

Checkpoint based recovery

Conclusion

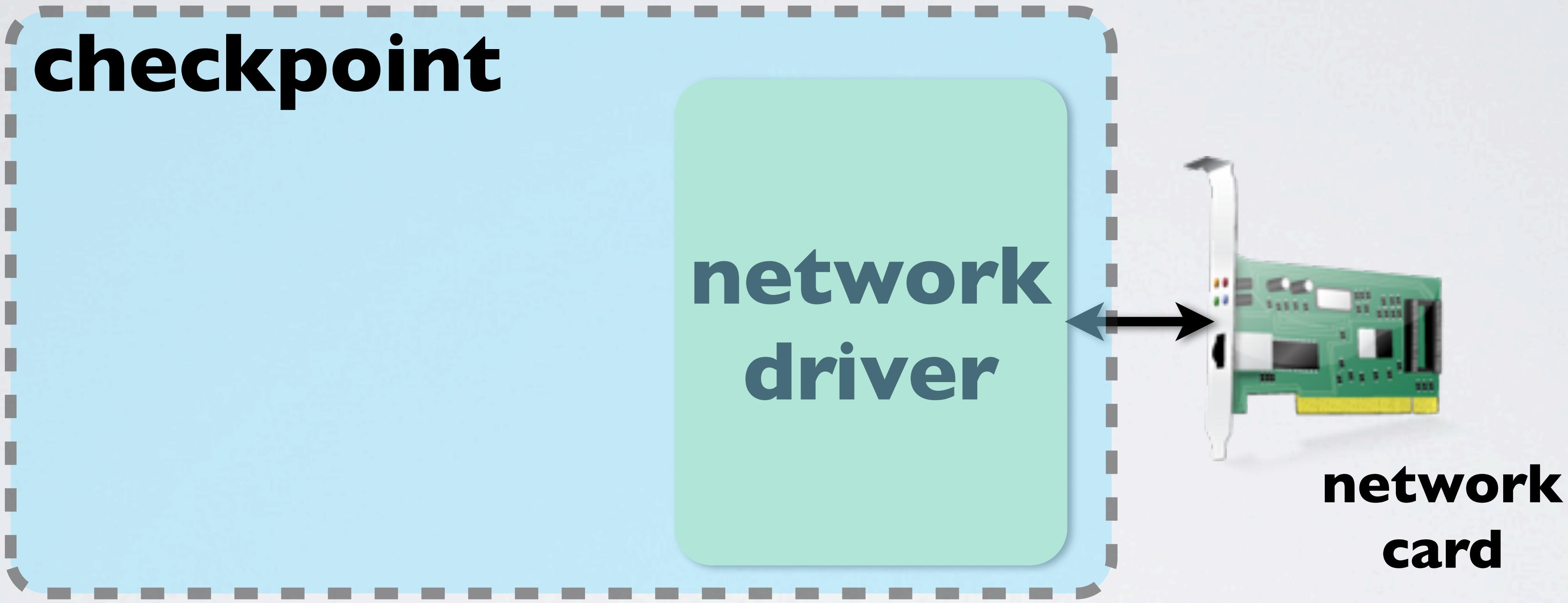
Checkpointing drivers is hard

★ **Existing mechanisms limited to capturing memory state**



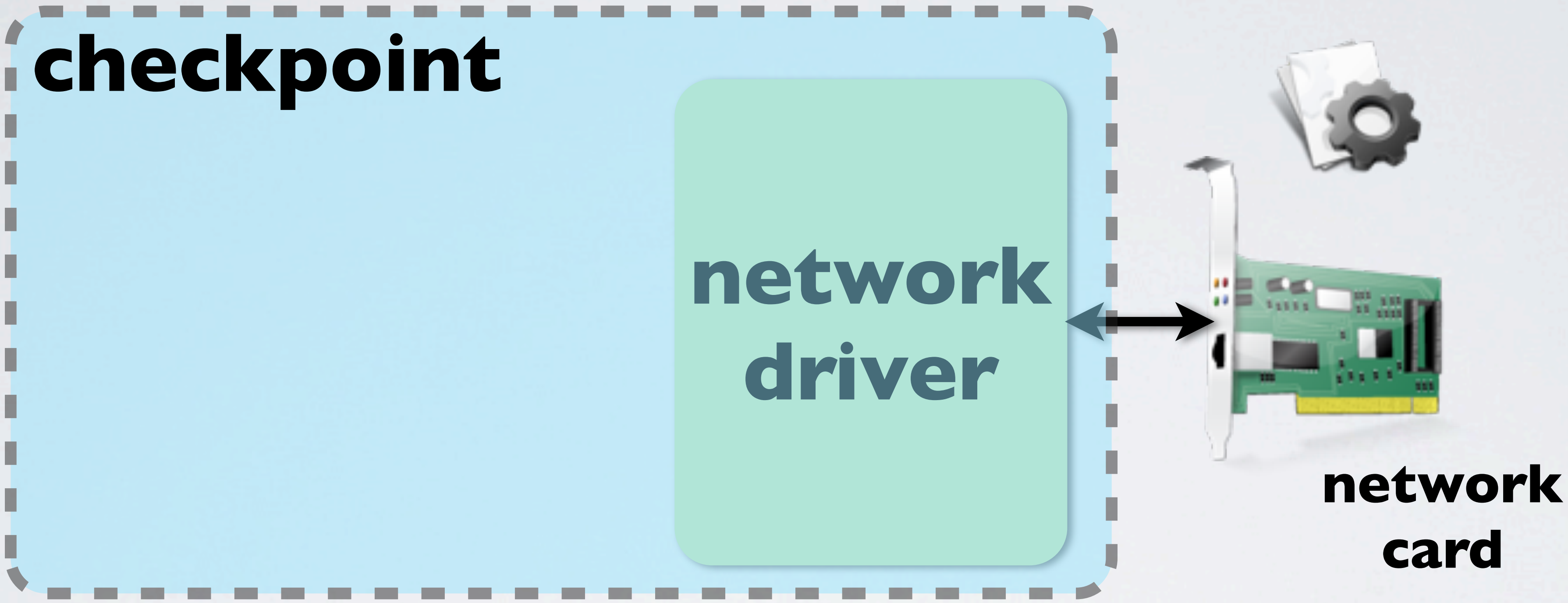
Checkpointing drivers is hard

★ **Existing mechanisms limited to capturing memory state**



Checkpointing drivers is hard

★ **Existing mechanisms limited to capturing memory state**

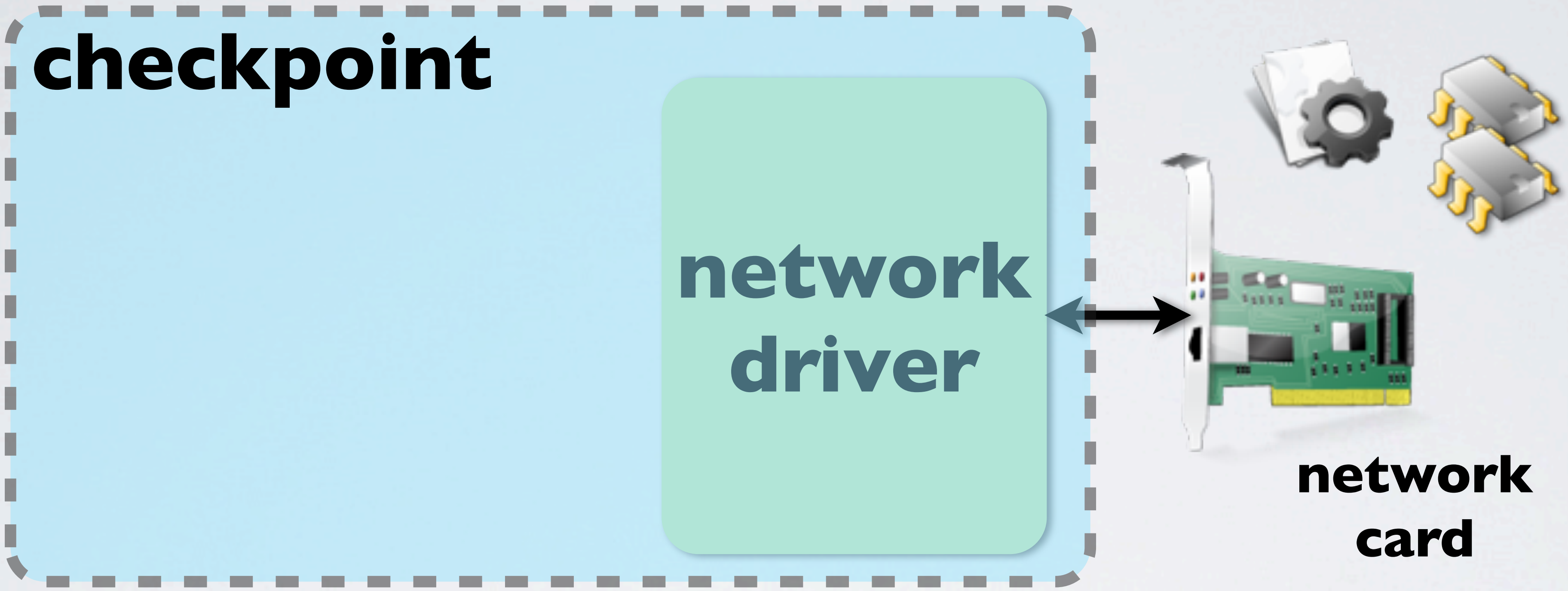


★ **Device state is not captured**

- ★ **Device configuration space**

Checkpointing drivers is hard

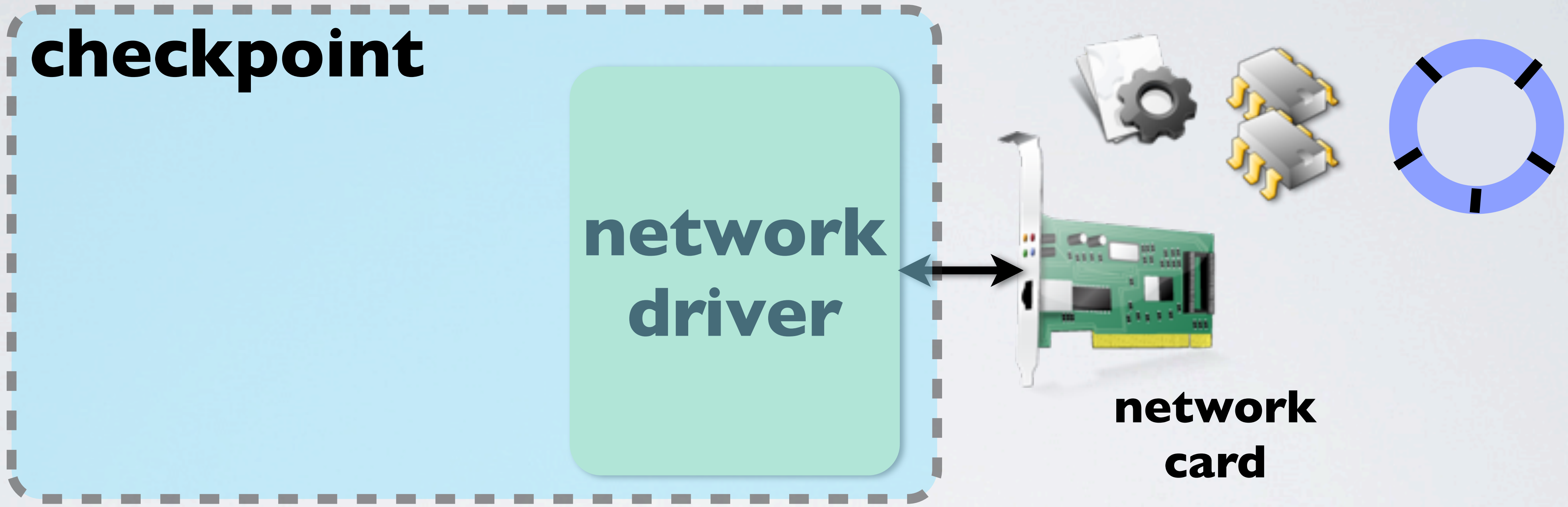
- ★ **Existing mechanisms limited to capturing memory state**



- ★ **Device state is not captured**
  - ★ **Device configuration space**
  - ★ **Internal device registers and counters**

Checkpointing drivers is hard

★ **Existing mechanisms limited to capturing memory state**

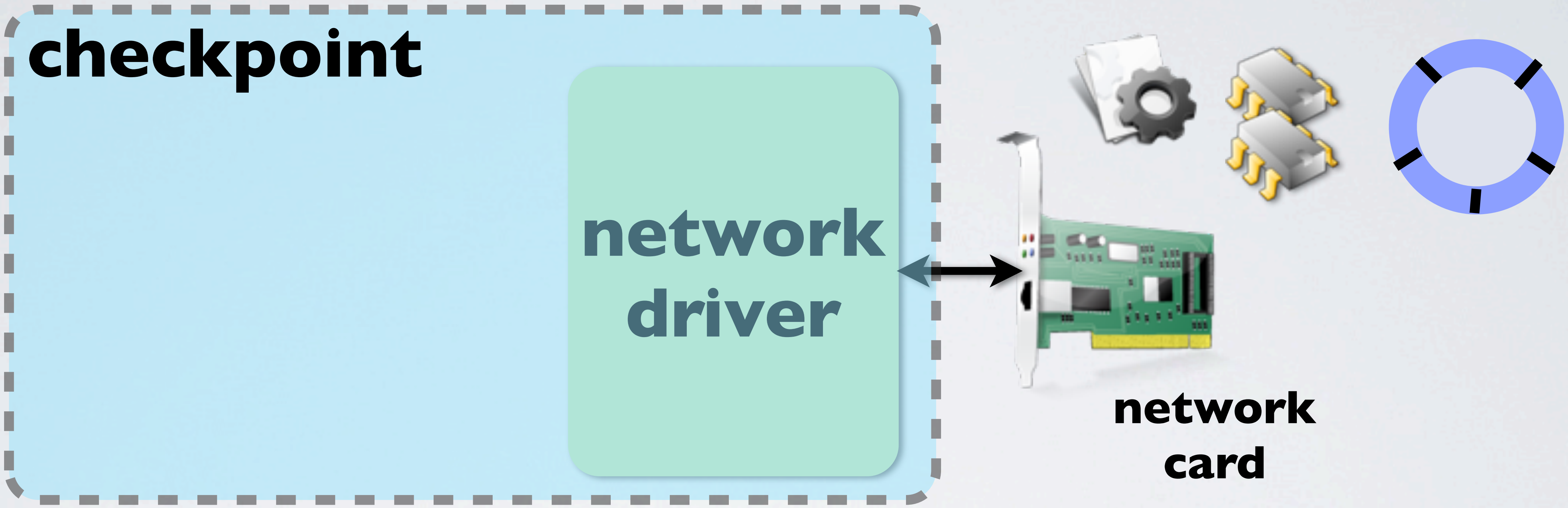


★ **Device state is not captured**

- ★ **Device configuration space**
- ★ **Internal device registers and counters**
- ★ **Memory buffer addresses used for DMA**

Checkpointing drivers is hard

★ Existing mechanisms limited to capturing **memory** state



★ Device state is not captured

- ★ Device configuration space
- ★ Internal device registers and counters
- ★ Memory buffer addresses used for DMA
- ★ Unique for every class, bus and vendor

Device checkpoint/restore from PM code

Suspend	Resume
Save config state	Restore config state
Save register state	Restore register state
Disable device	Restore s/w state & reset
Copy-out s/w state	Re-attach/Enable device
Suspend device	Device Ready

Device checkpoint/restore from PM code

Suspend	Resume
Save config state	Restore config state
Save register state	Restore register state
	Restore s/w state & reset
Copy-out s/w state	Re-attach/Enable device
Suspend device	Device Ready

Device checkpoint/restore from PM code

Suspend

- Save config state
- Save register state
- Copy-out s/w state

Resume

- Restore config state
- Restore register state
- Restore s/w state & reset
- Re-attach/Enable device
- Device Ready

Device checkpoint/restore from PM code

Suspend

- Save config state
- Save register state
- Copy-out s/w state

Resume

- Restore config state
- Restore register state
- Restore s/w state & reset
- Re-attach/Enable device
- Device Ready

Device checkpoint/restore from PM code

Checkpoint	Resume
Save config state	Restore config state
Save register state	Restore register state
Copy-out s/w state	Restore s/w state & reset
	Re-attach/Enable device
	Device Ready

Device checkpoint/restore from PM code

Checkpoint

- Save config state
- Save register state
- Copy-out s/w state

Resume

- Restore config state
- Restore register state
- Restore s/w state & reset
- Re-attach/Enable device

Device checkpoint/restore from PM code

Checkpoint

- Save config state
- Save register state
- Copy-out s/w state

Resume

- Restore config state
- Restore register state
- Restore s/w state & reset

Device checkpoint/restore from PM code

Checkpoint

Checkpoint
Save config state
Save register state
Copy-out s/w state

Restore

Restore
Restore config state
Restore register state
Restore s/w state & reset

Device checkpoint/restore from PM code

Checkpoint

Save config state
Save register state
Copy-out s/w state

Restore

Restore config state
Restore register state
Restore s/w state & reset

Suspend/resume code provides device checkpoint functionality

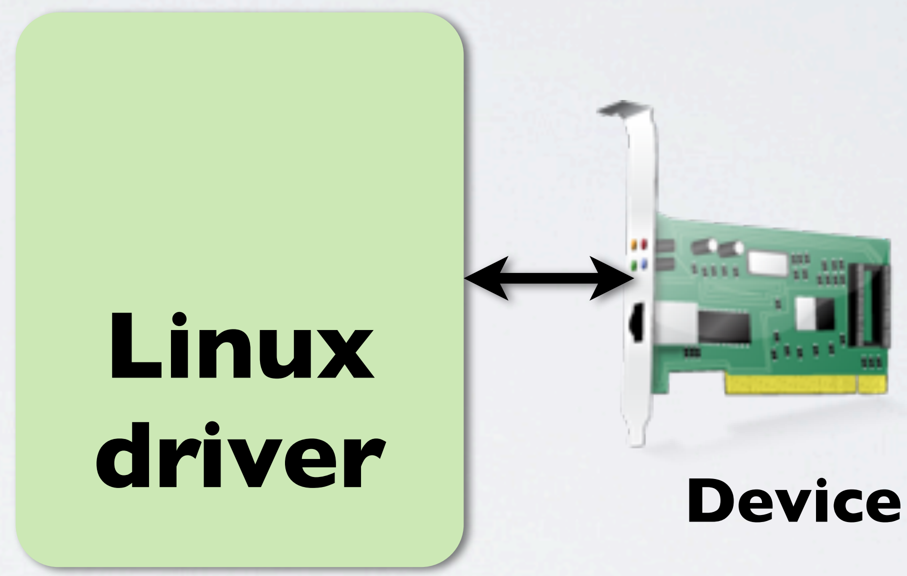
Intuition with power management

Intuition with power management

- **Intuition: Power management code captures device specific state for every driver**
  - **Our study: Present in 76% of all common classes**

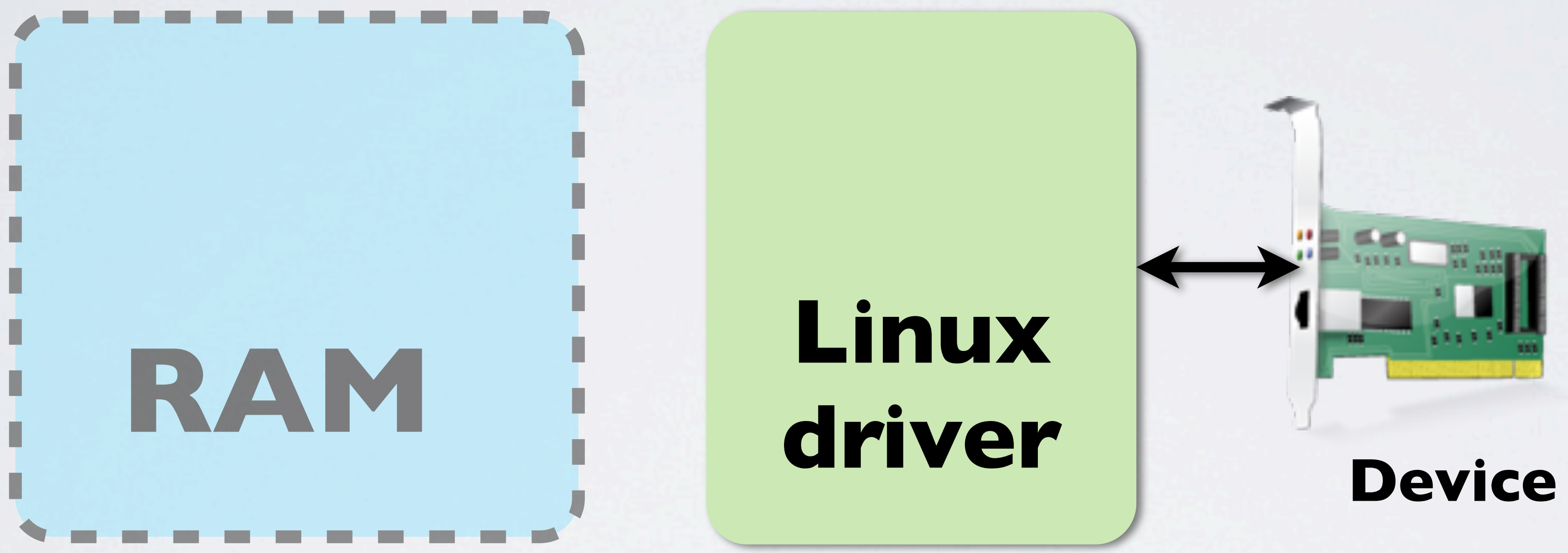
Intuition with power management

- **Intuition: Power management code captures device specific state for every driver**
  - **Our study: Present in 76% of all common classes**



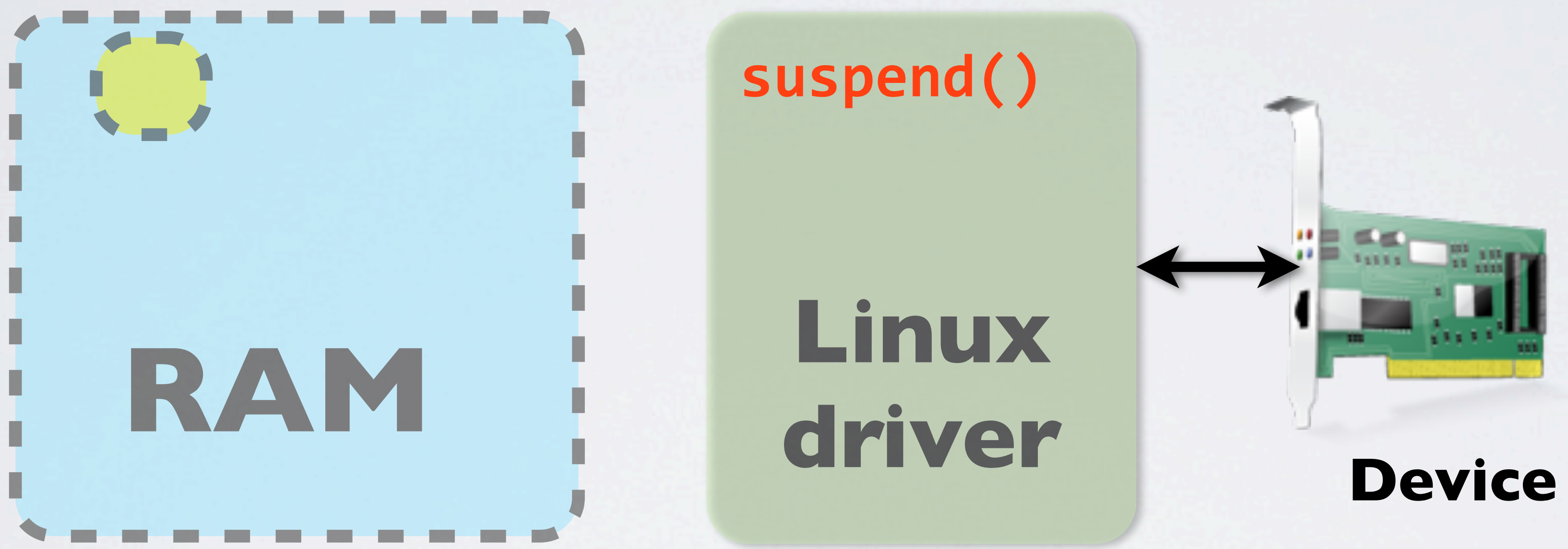
Intuition with power management

- **Intuition: Power management code captures device specific state for every driver**
  - **Our study: Present in 76% of all common classes**



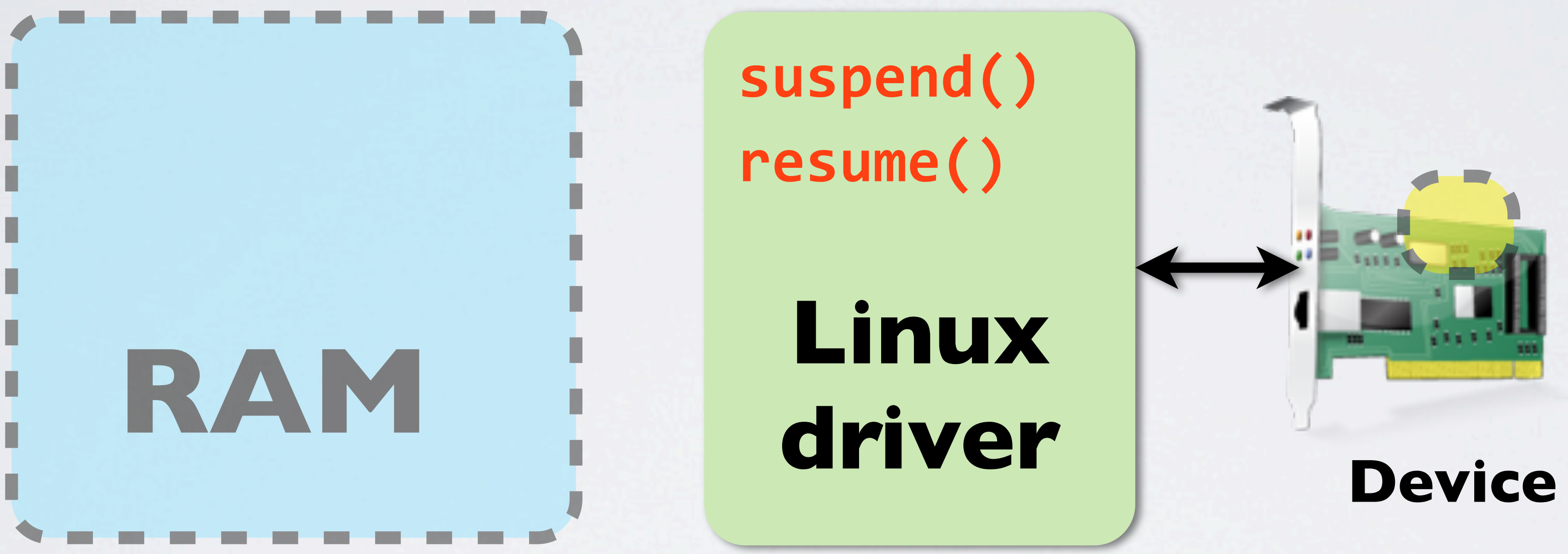
Intuition with power management

- **Intuition: Power management code captures device specific state for every driver**
  - **Our study: Present in 76% of all common classes**



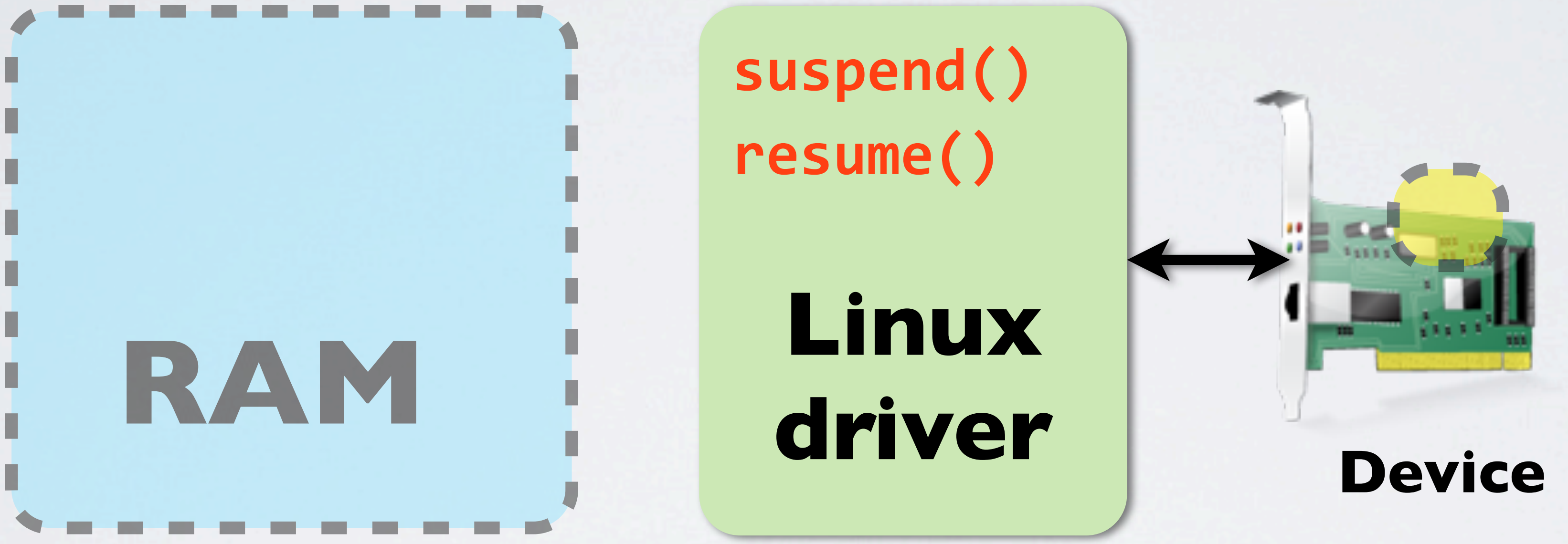
Intuition with power management

- **Intuition: Power management code captures device specific state for every driver**
  - **Our study: Present in 76% of all common classes**



Intuition with power management

- **Intuition: Power management code captures device specific state for every driver**
  - **Our study: Present in 76% of all common classes**



- **Refactor power management code for device checkpoints**
  - **Correct: Developer captures unique device semantics**
  - **Fast: Avoids probe and latency critical for applications**

Synergy of isolation and recovery

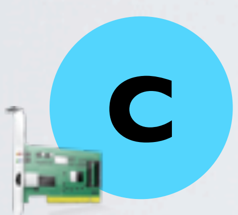
- **Goal: Improve driver recovery with minor changes to drivers**
- **Solution: Run drivers as transactions using device checkpoints**

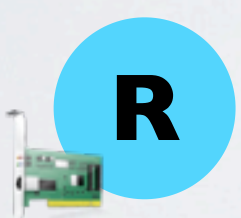
Synergy of isolation and recovery

- **Goal: Improve driver recovery with minor changes to drivers**
- **Solution: Run drivers as transactions using device checkpoints**

Device state

- **Developers export checkpoint/restore in drivers**





Synergy of isolation and recovery

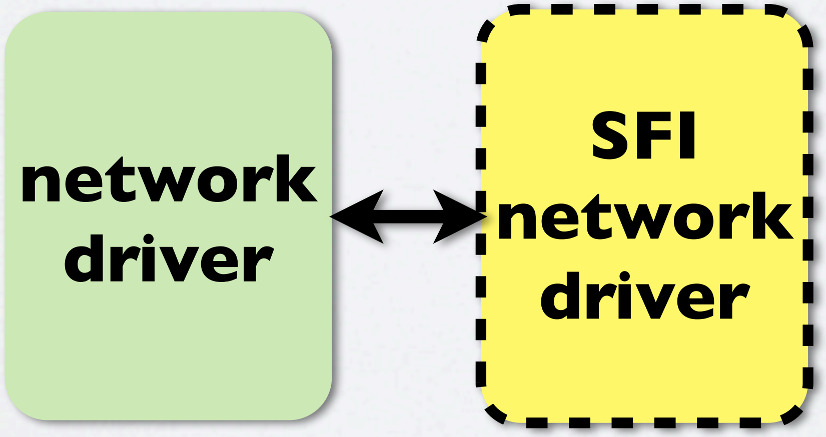
- **Goal: Improve driver recovery with minor changes to drivers**
- **Solution: Run drivers as transactions using device checkpoints**

Device state

- **Developers export checkpoint/restore in drivers**

Driver state

- **Run drivers invocations as memory transactions**
- **Use source transformation to copy parameters and run on separate stack**



Synergy of isolation and recovery

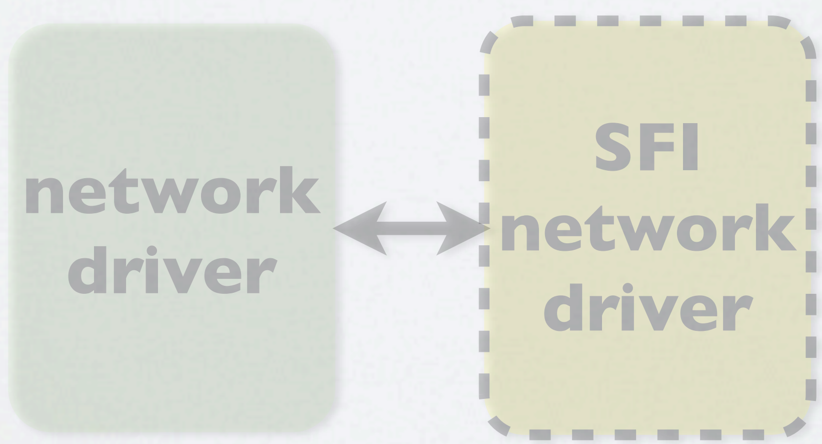
- ★ **Goal: Improve driver recovery with minor changes to drivers**
- ★ **Solution: Run drivers as transactions using device checkpoints**

Device state

- ★ **Developers export checkpoint/restore in drivers**

Driver state

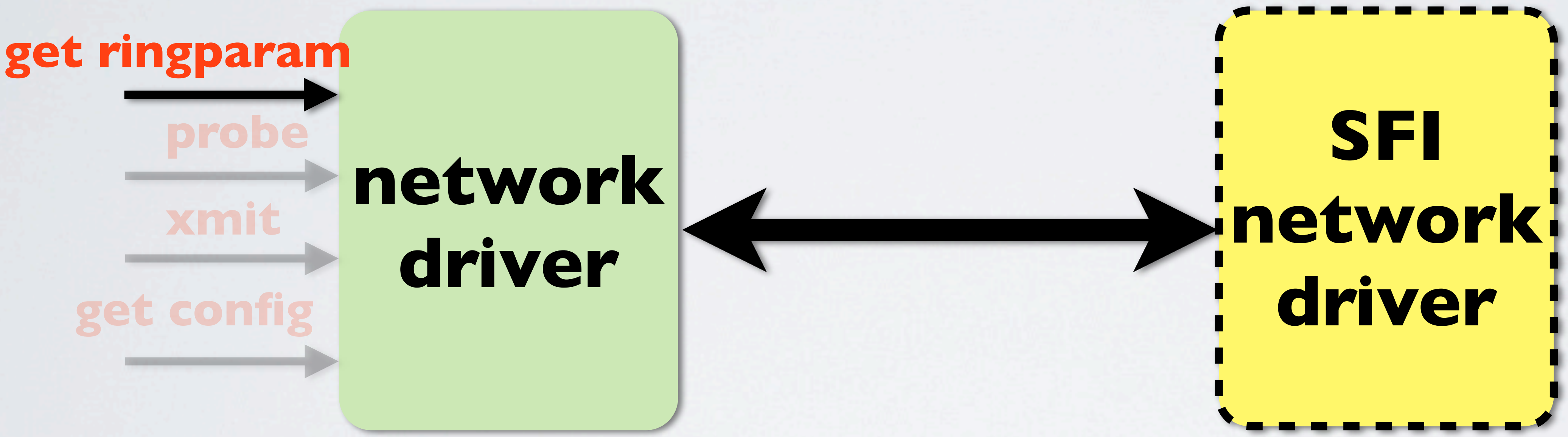
- ★ **Run drivers invocations as memory transactions**
- ★ **Use source transformation to copy parameters and run on separate stack**



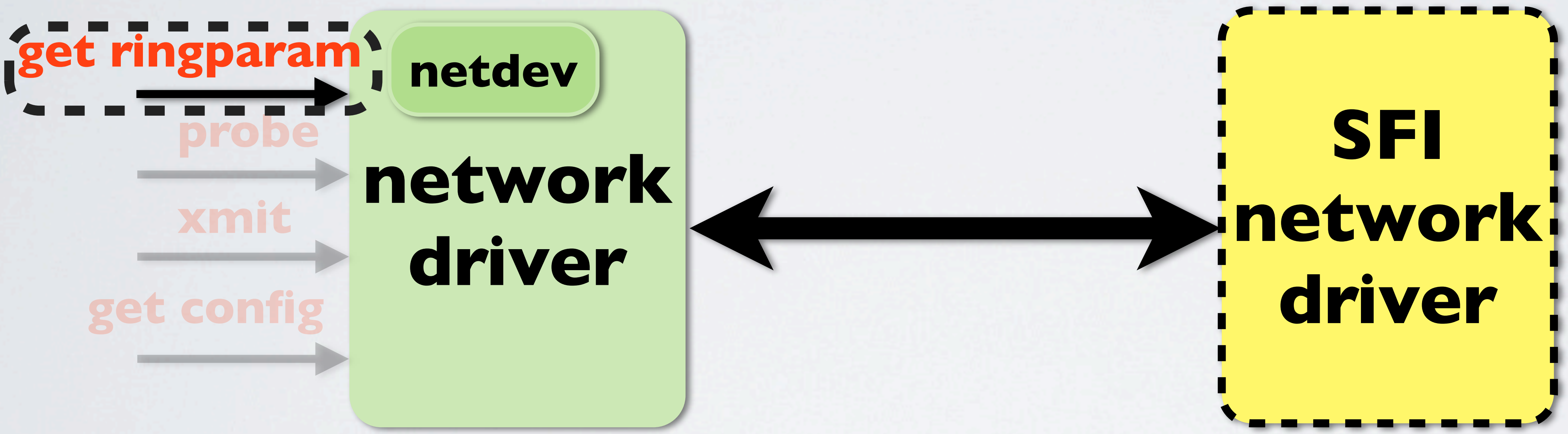
Execution model

- ★ **Checkpoint device**
- ★ **Execute driver code as memory transactions**
- ★ **On failure, rollback and restore device**
- ★ **Re-use existing device locks in the driver**

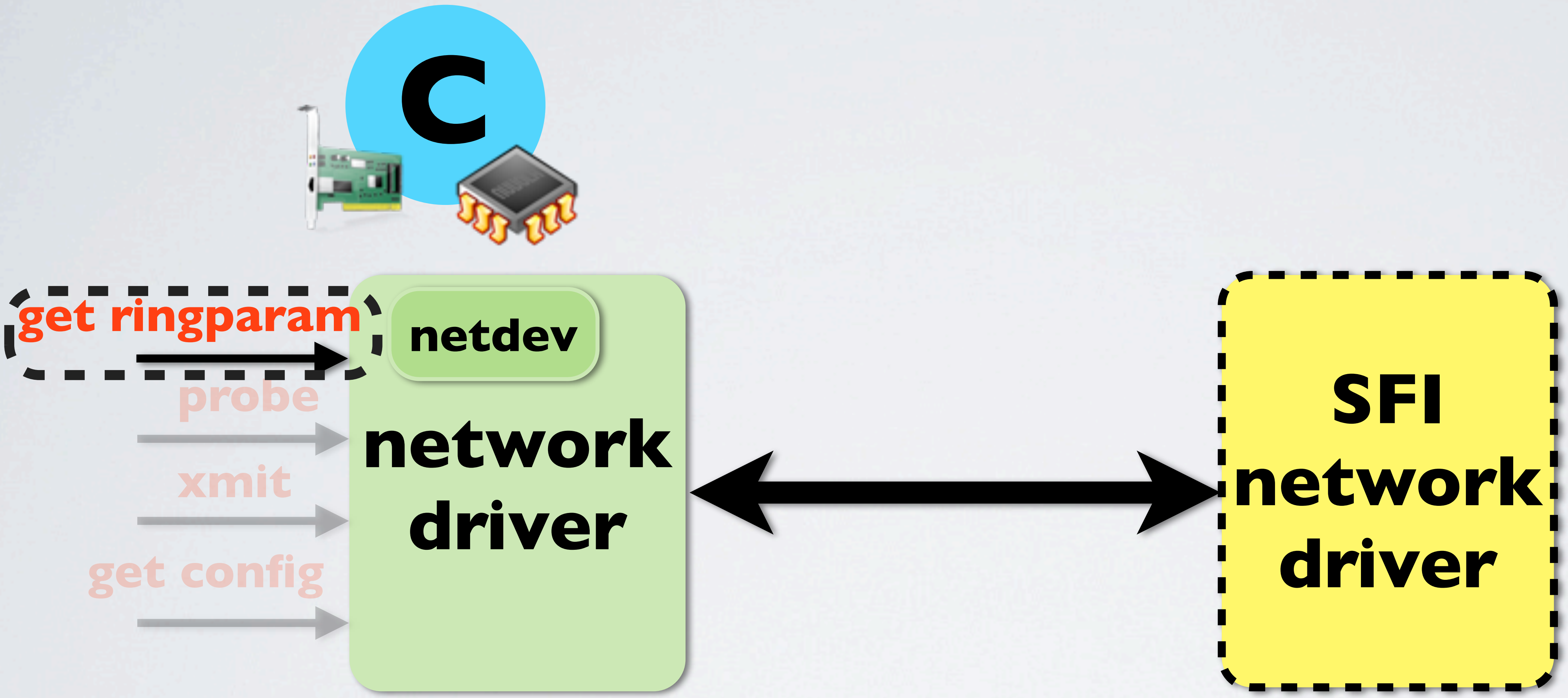
Example transactional execution



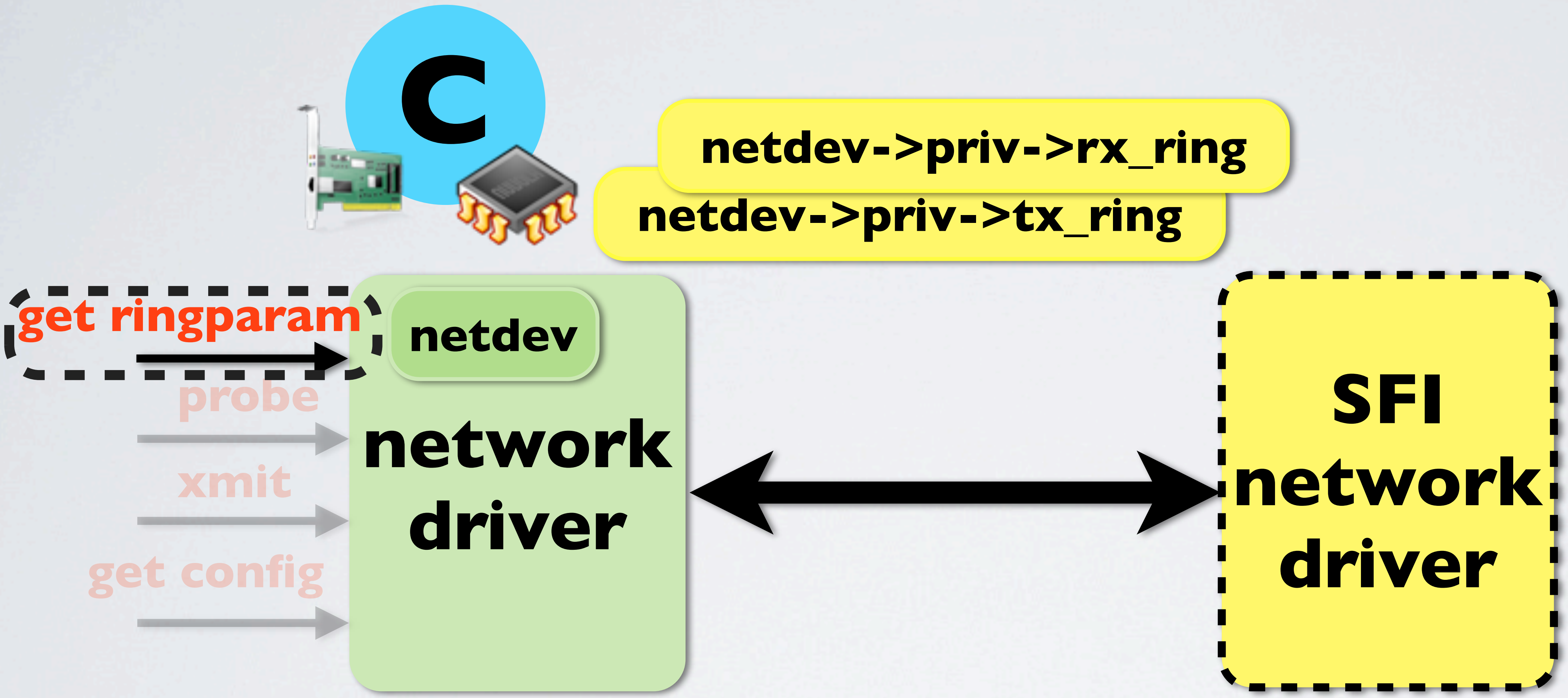
Example transactional execution



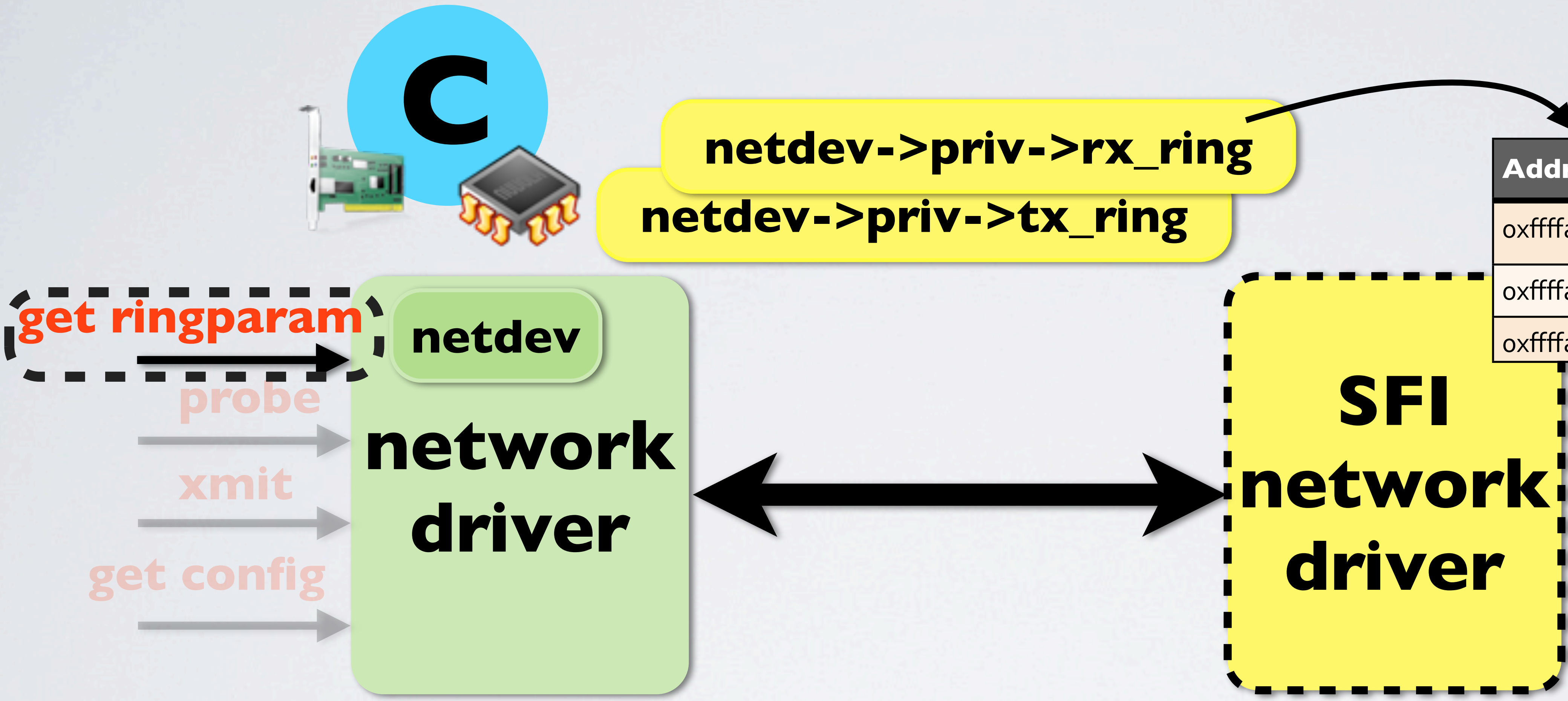
Example transactional execution



Example transactional execution



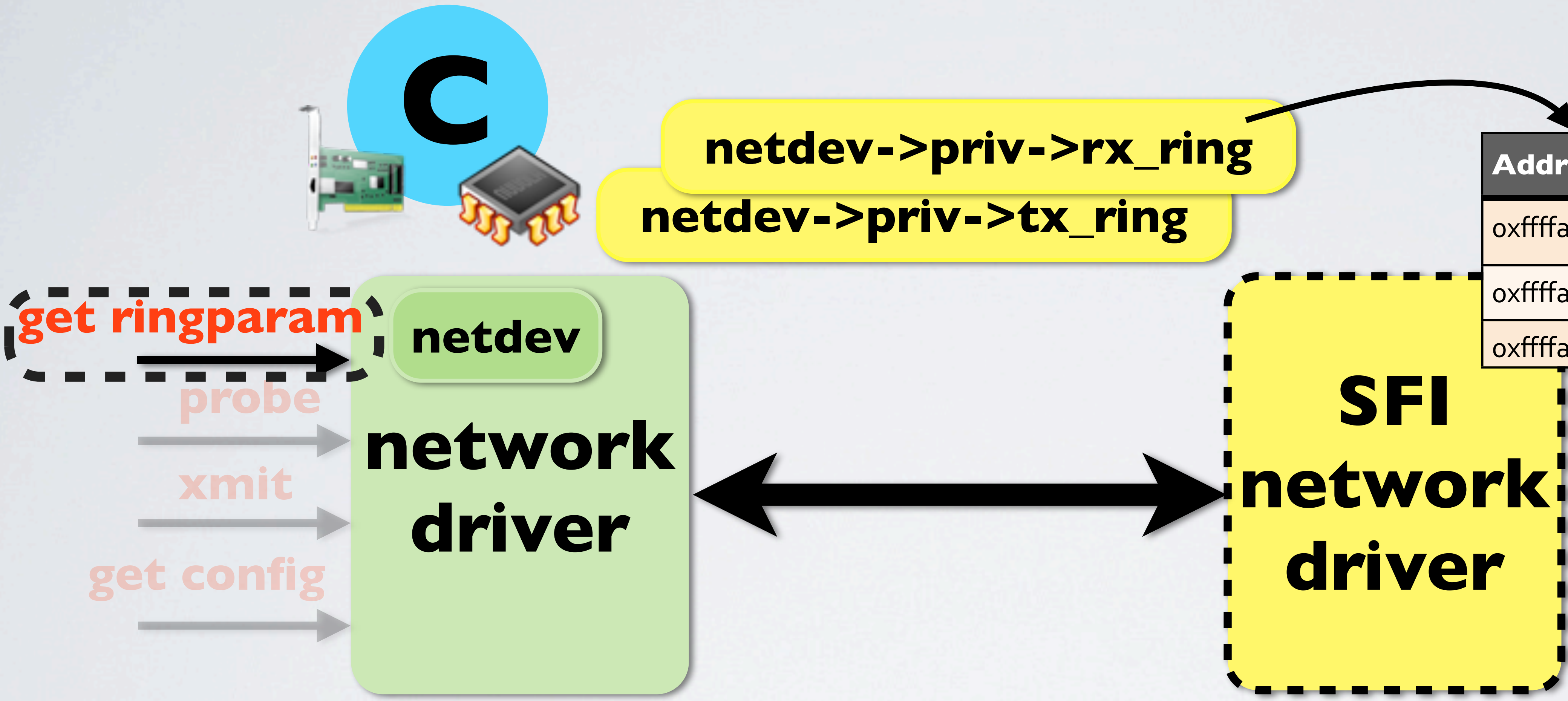
Example transactional execution



Range Table

Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

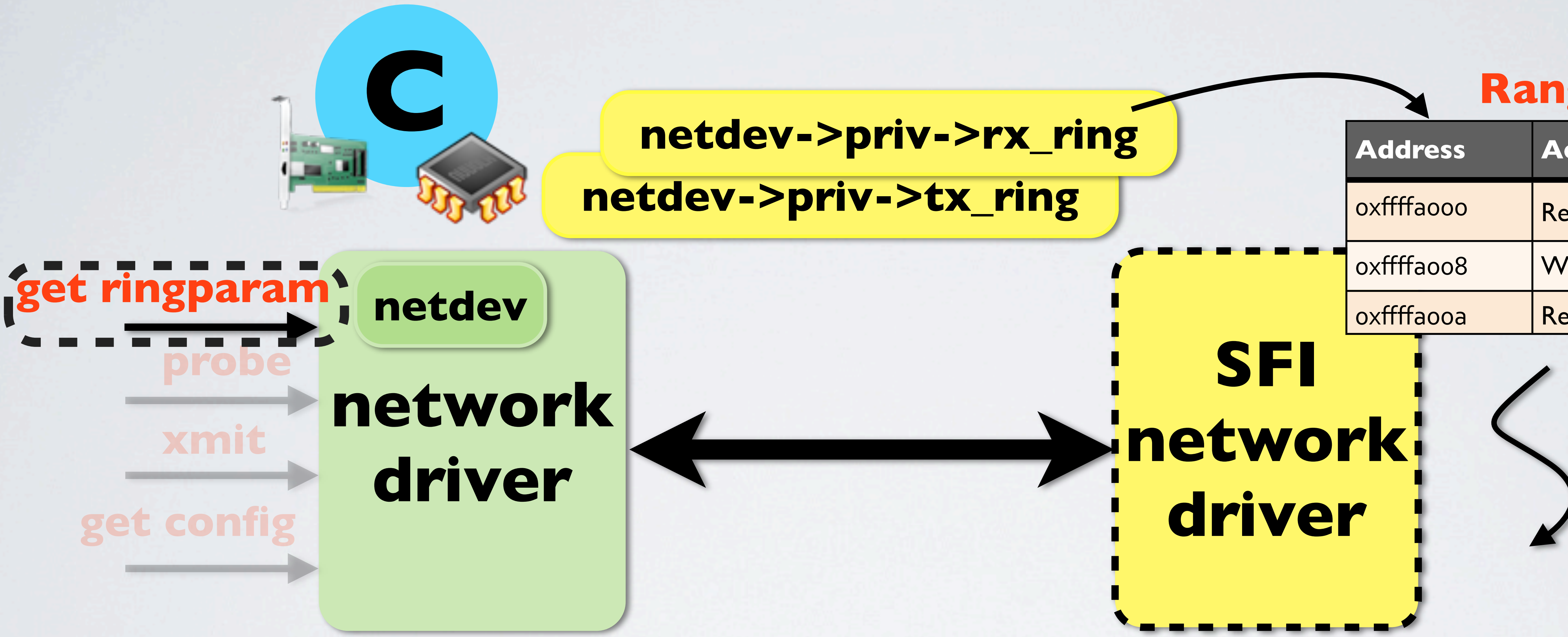
Example transactional execution



Range Table

Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

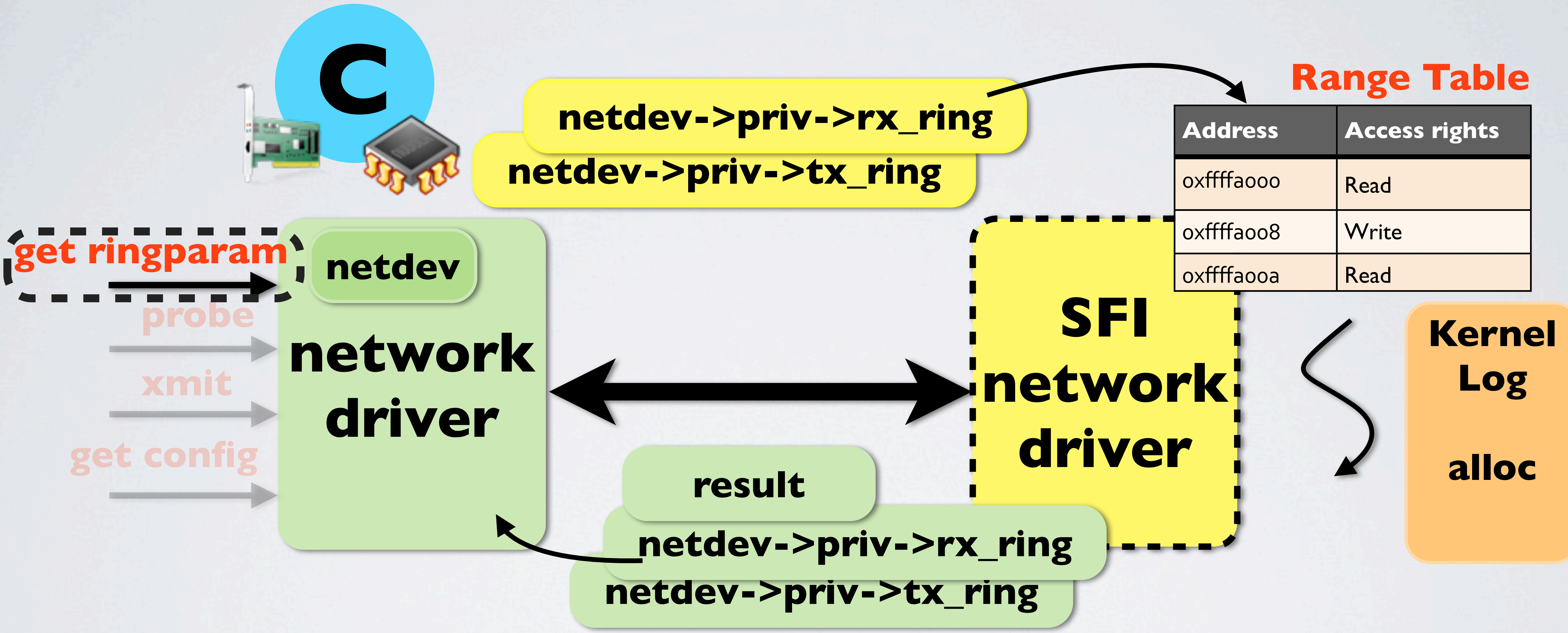
Example transactional execution



Range Table

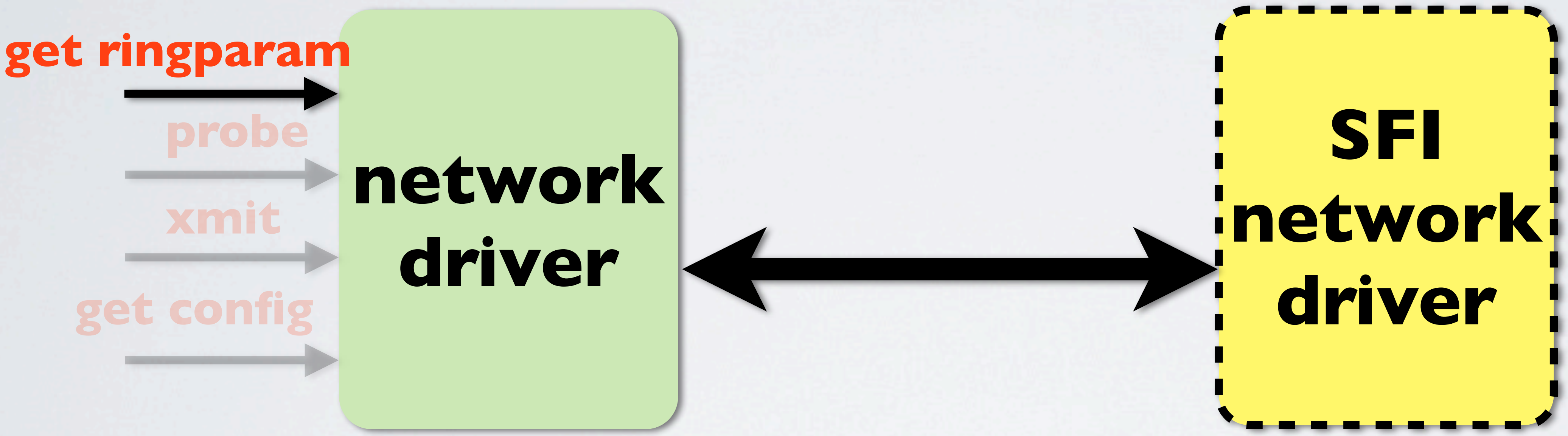
Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

Example transactional execution

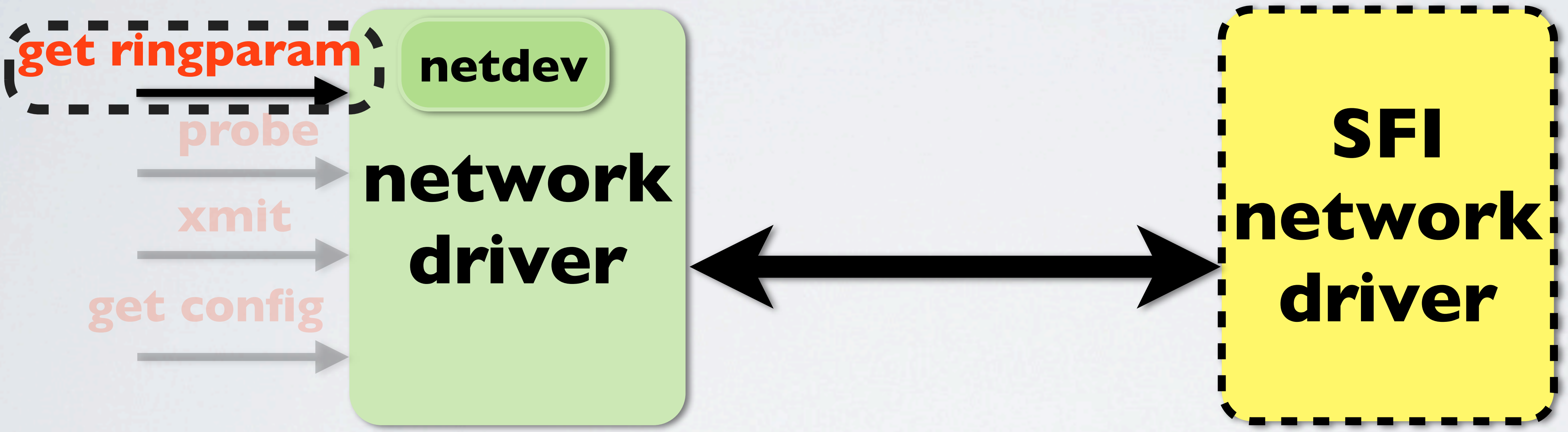


Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

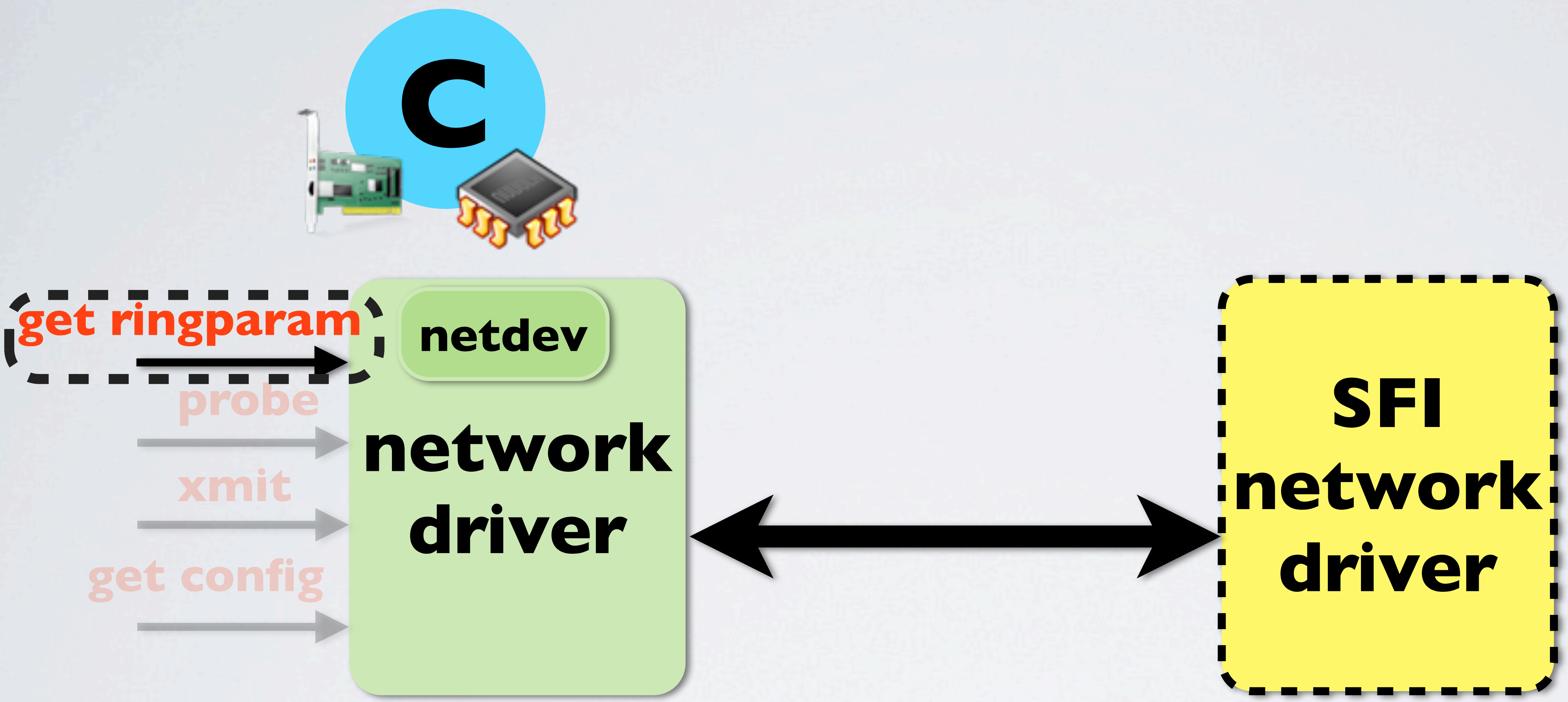
Example failed transaction



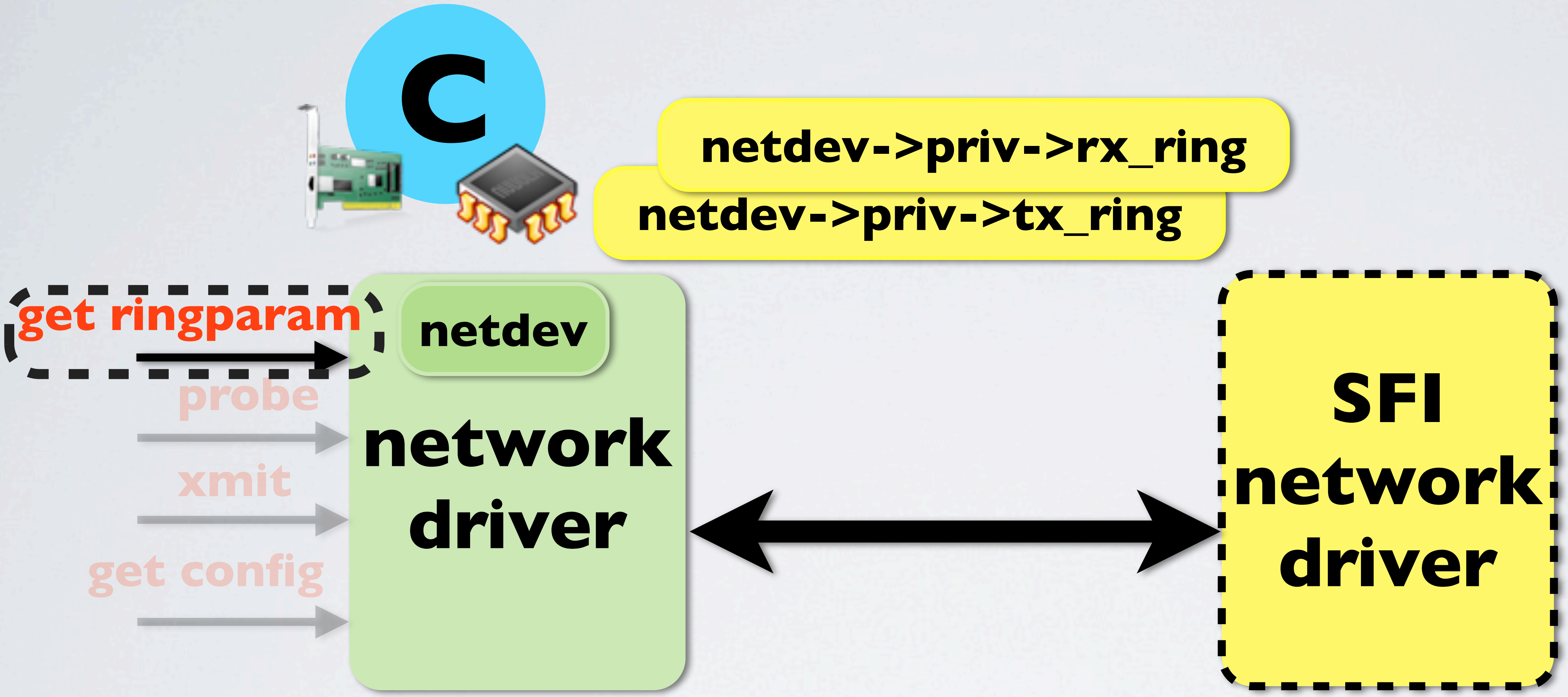
Example failed transaction



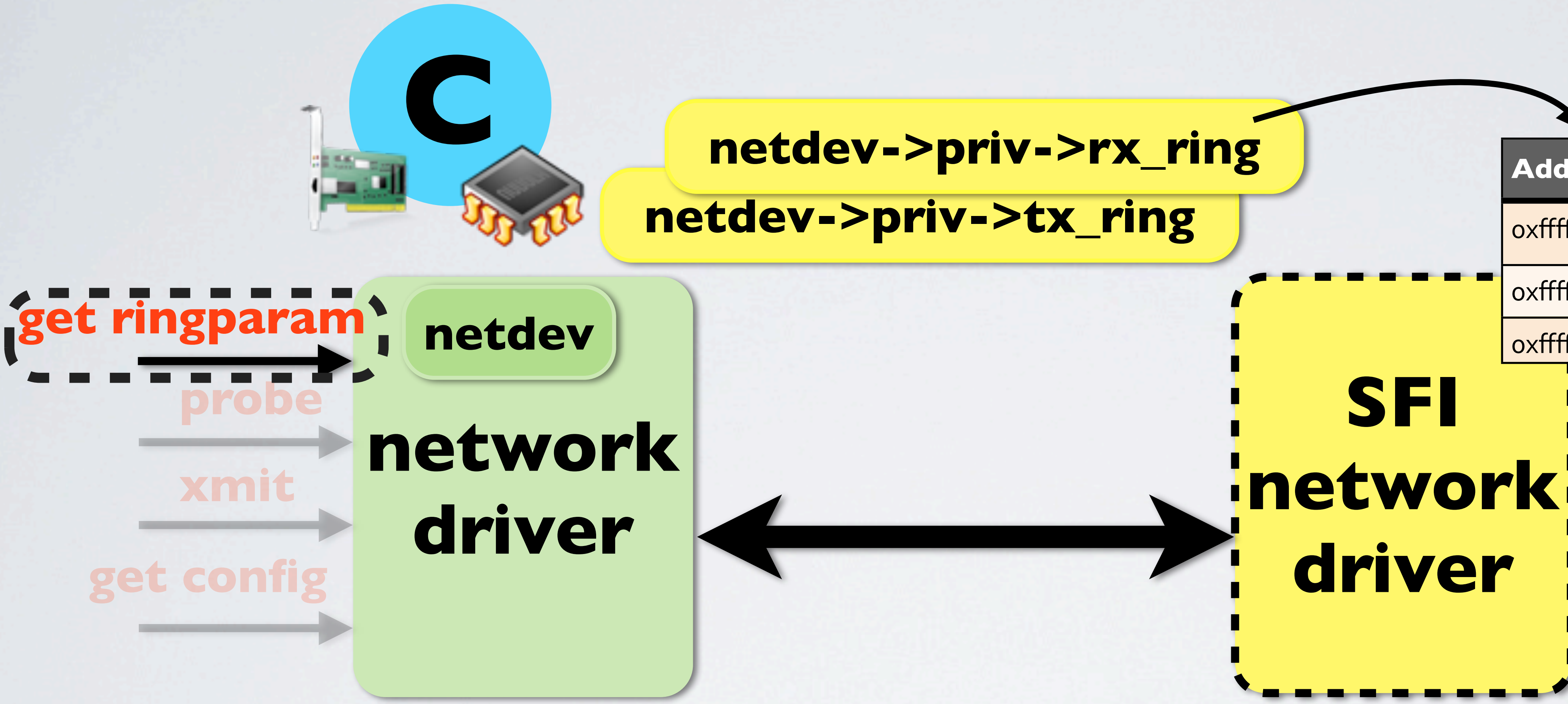
Example failed transaction



Example failed transaction



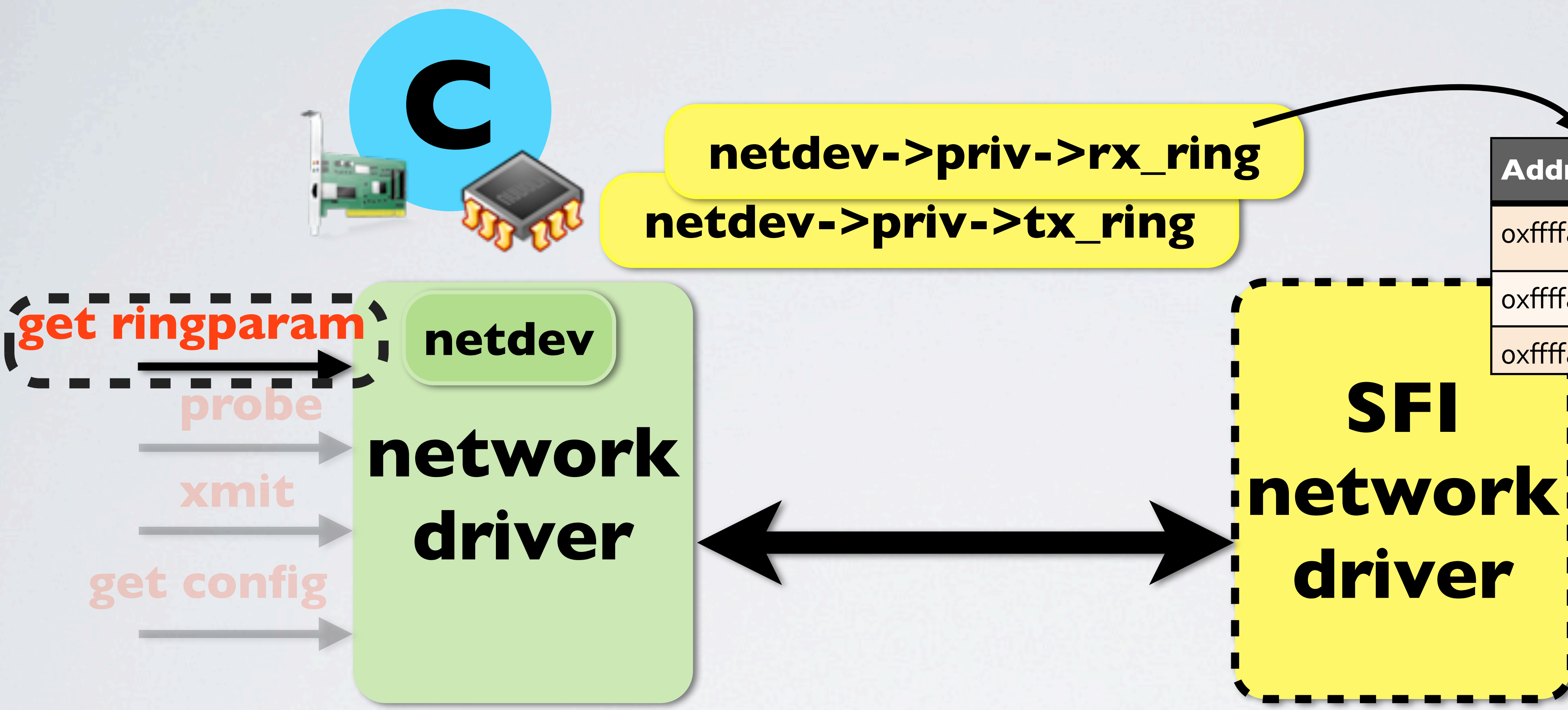
Example failed transaction



Range Table

Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

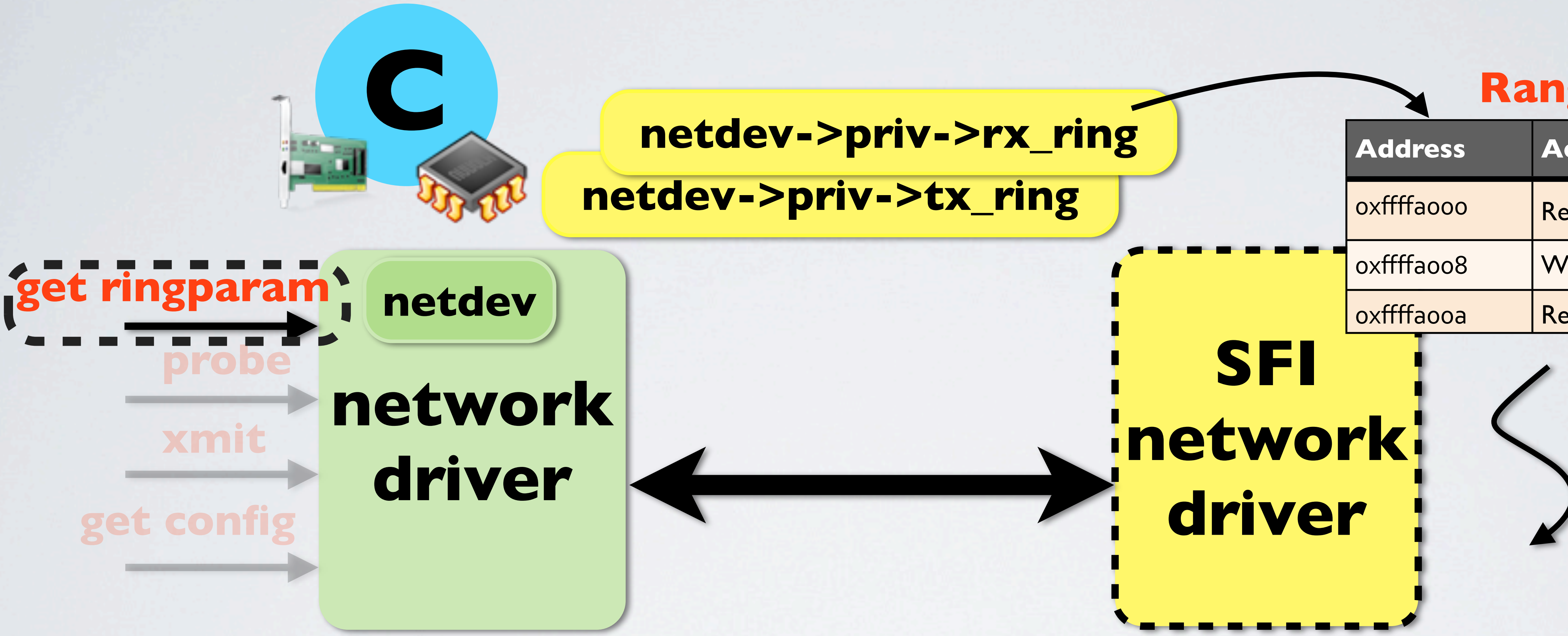
Example failed transaction



Range Table

Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

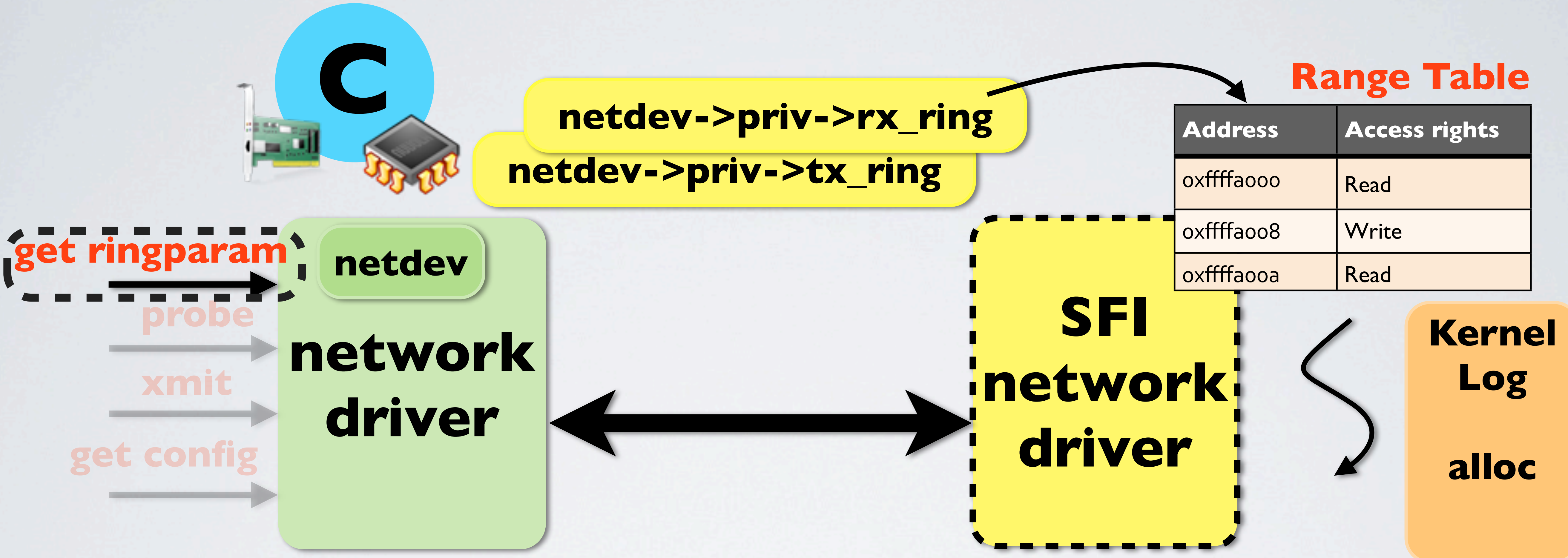
Example failed transaction



Range Table

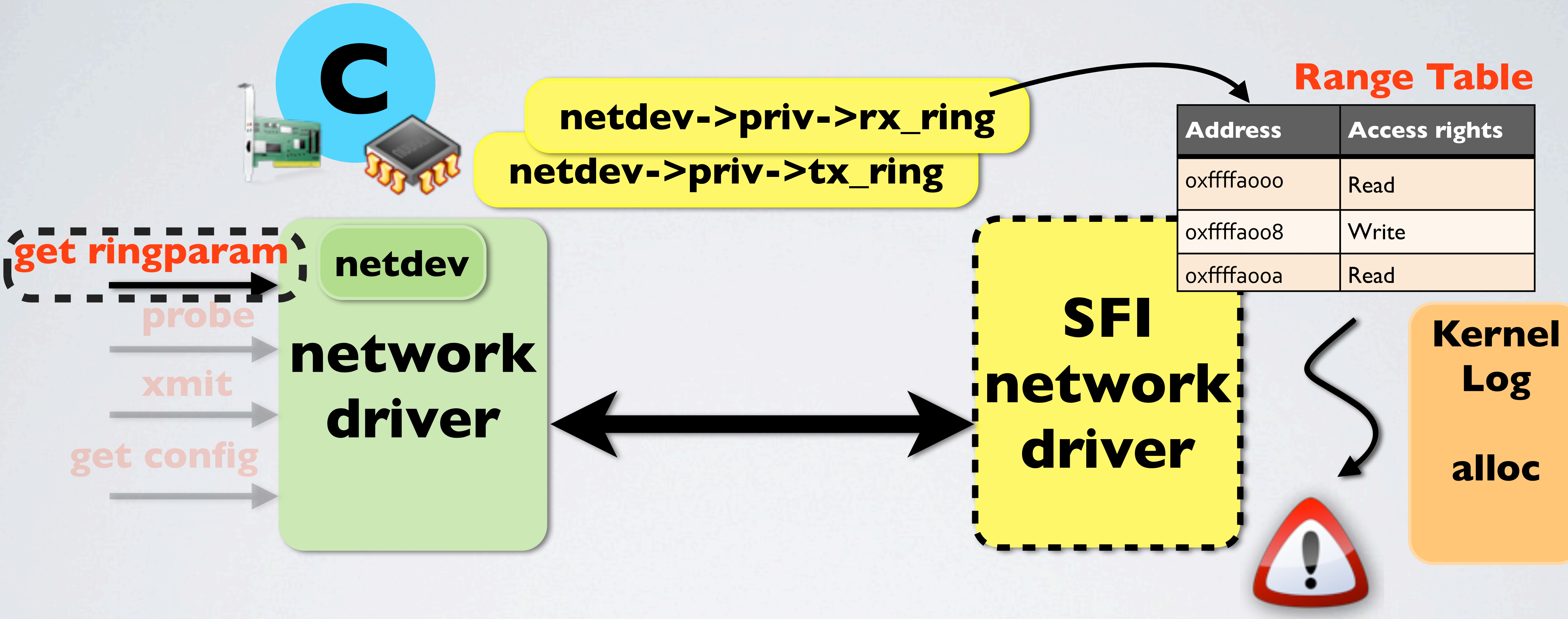
Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

Example failed transaction



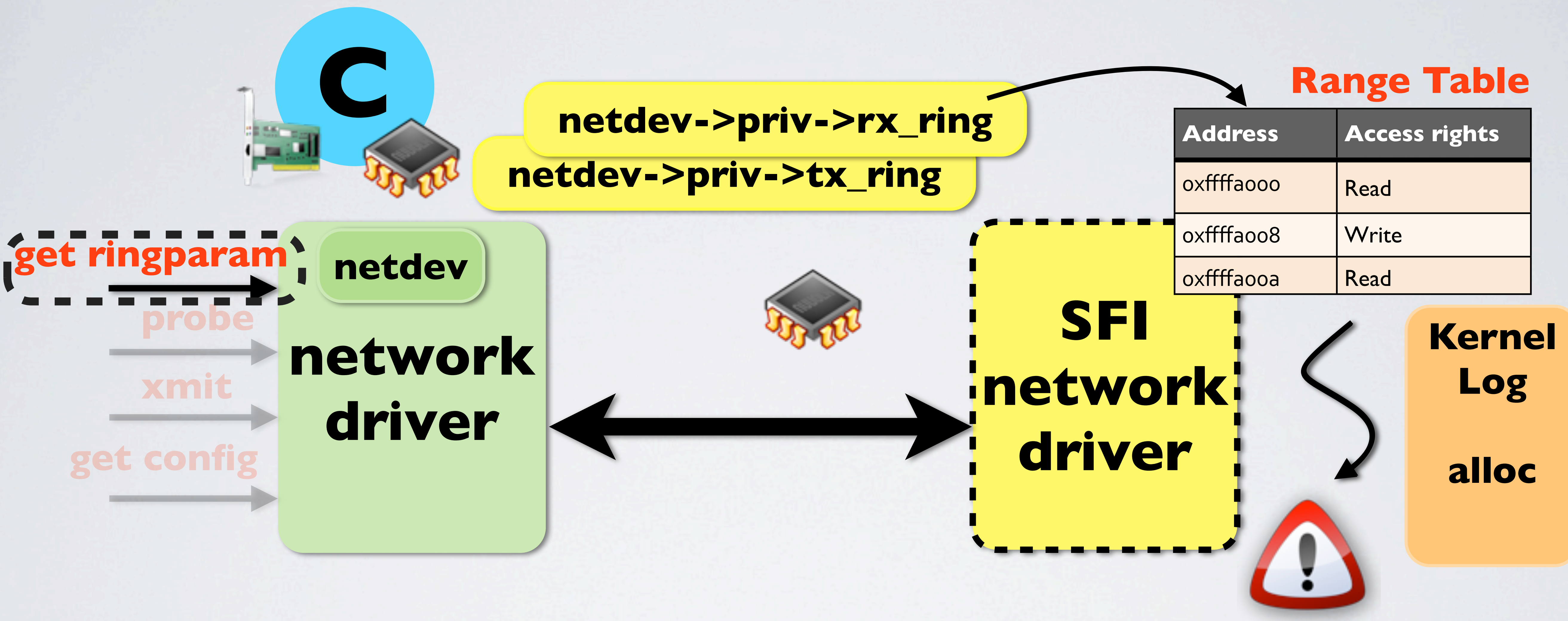
Address	Access rights
0xffffa000	Read
0xffffa008	Write
0xffffa00a	Read

Example failed transaction

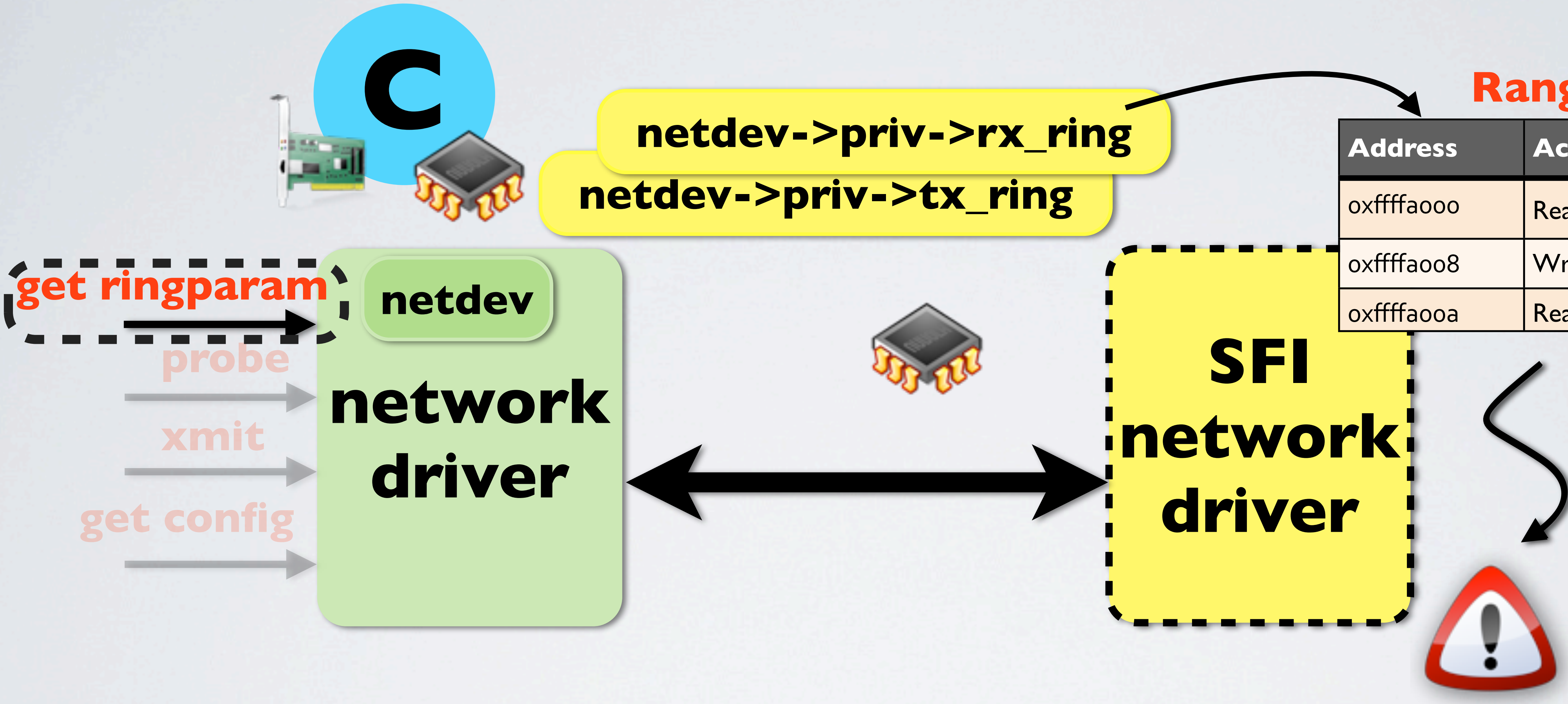


Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

Example failed transaction



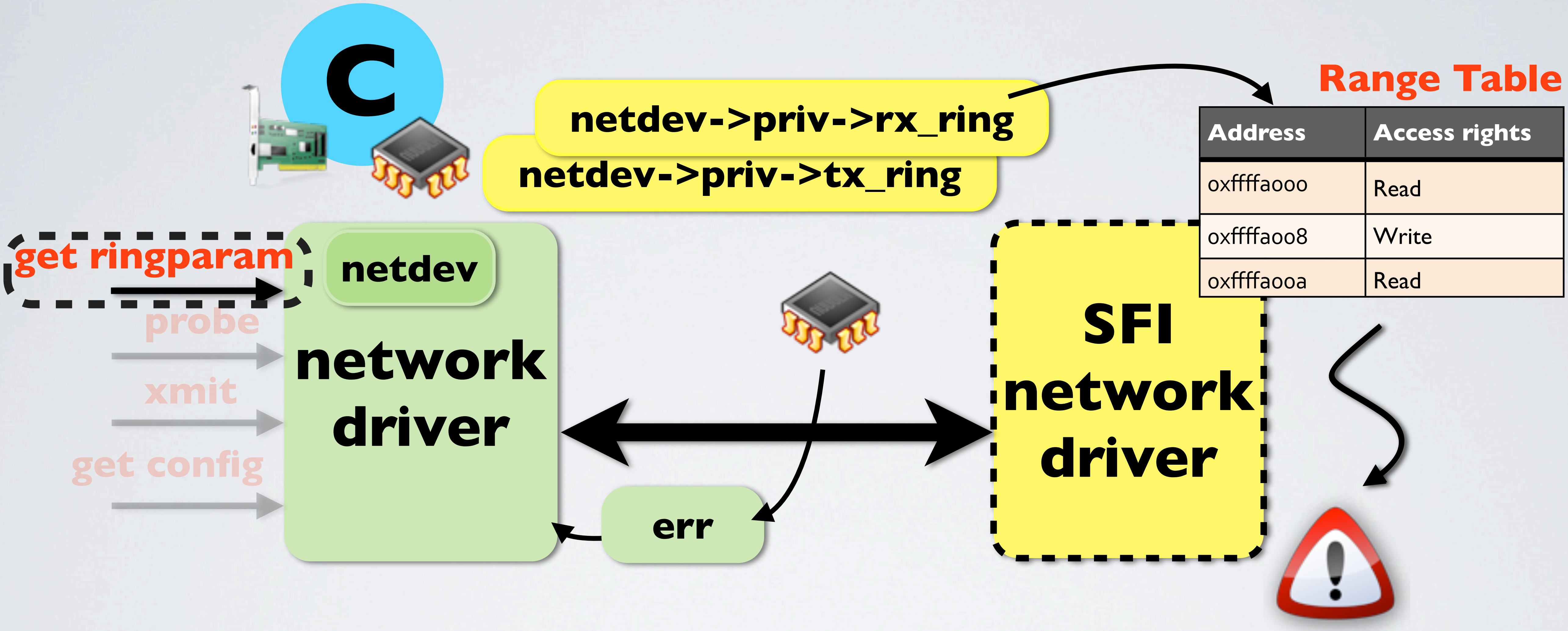
Example failed transaction



Range Table

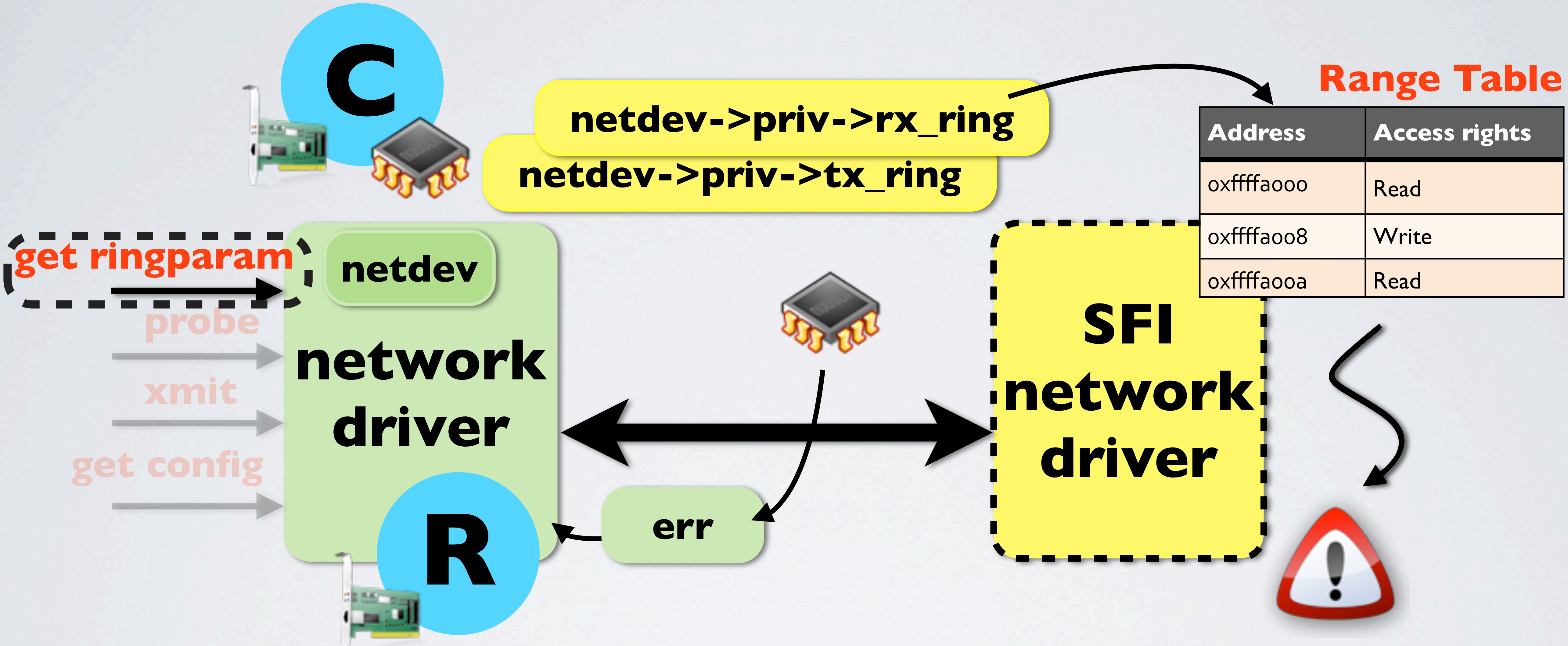
Address	Access rights
0xffffa000	Read
0xffffa008	Write
0xffffa00a	Read

Example failed transaction



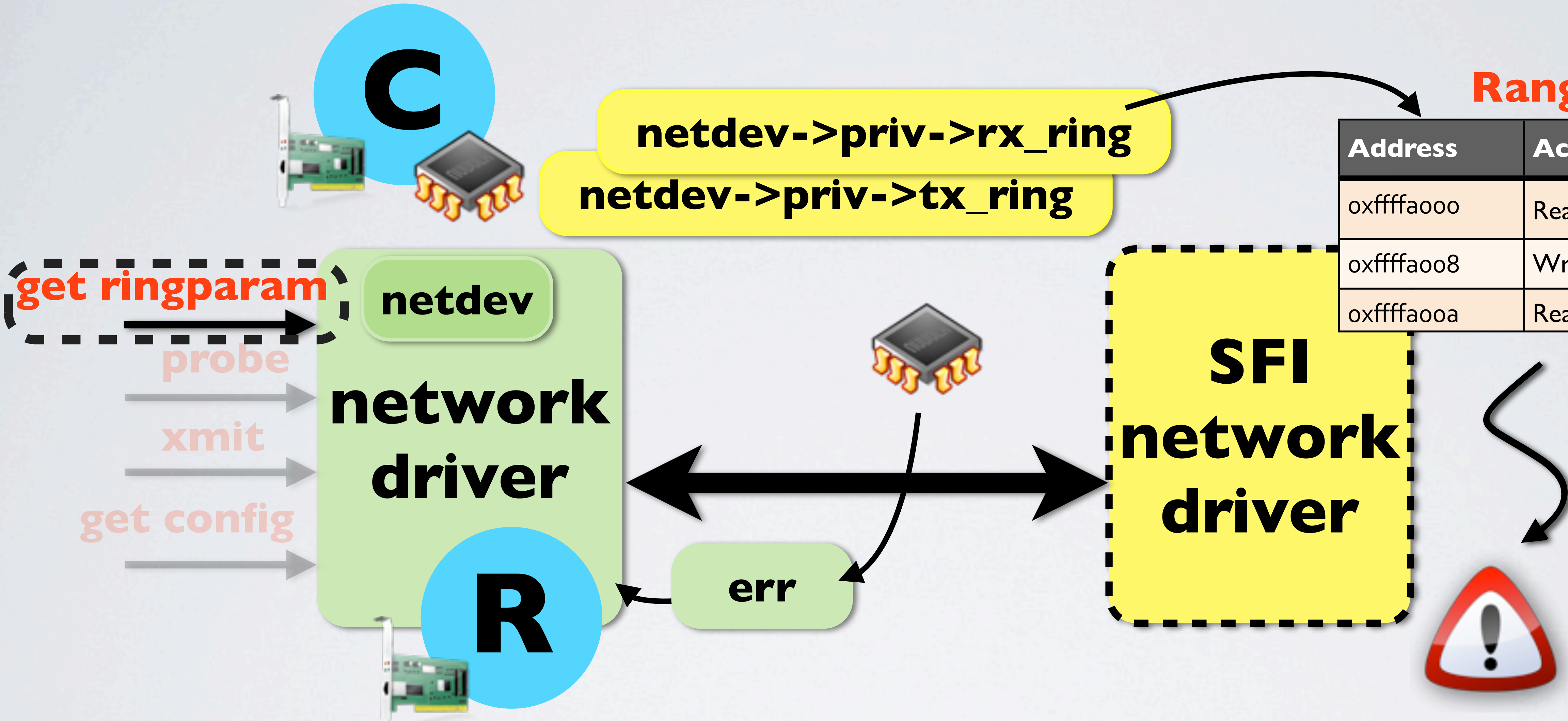
Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

Example failed transaction



Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

Example failed transaction



Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

FGFT provides transactional execution of driver entry points

Outline

Introduction

Fine-grained isolation

Checkpoint based recovery

Evaluation & Conclusions

Outline

Introduction

Fine-grained isolation

Checkpoint based recovery

Evaluation & Conclusion

Recovery speedup

Driver	Class	Bus	Restart recovery	FGFT recovery	Speedup
8139too	net	PCI	0.31s	70µs	4400
e1000	net	PCI	1.80s	295ms	6
r8169	net	PCI	0.12s	40µs	3000
pegasus	net	USB	0.15s	5ms	30
ens1371	sound	PCI	1.03s	115ms	9
psmouse	input	serio	0.68s	410ms	1.65

FGFT provides significant speedup in driver recovery

Static and dynamic fault injection

Driver	Injected Faults	Benign Faults	Native Crashes	FGFT Crashes
8139too	43	0	43	NONE
e1000	47	0	47	NONE
r8169	36	0	36	NONE
pegasus	34	1	33	NONE
ens1371	22	1	21	NONE
psmouse	46	0	46	NONE
TOTAL	258	2	256	NONE

FGFT survives multiple static and dynamic faults

Programming effort

Driver	LOC	Isolation annotations		Recovery additions	
		Driver annotations	Kernel annotations	LOC Moved	LOC Added
8139too	1,904	15	20	26	4
e1000	13,973	32		32	10
r8169	2,993	10		17	5
pegasus	1,541	26	12	22	5
ens1371	2,110	23	66	16	6
psmouse	2,448	11	19	19	6

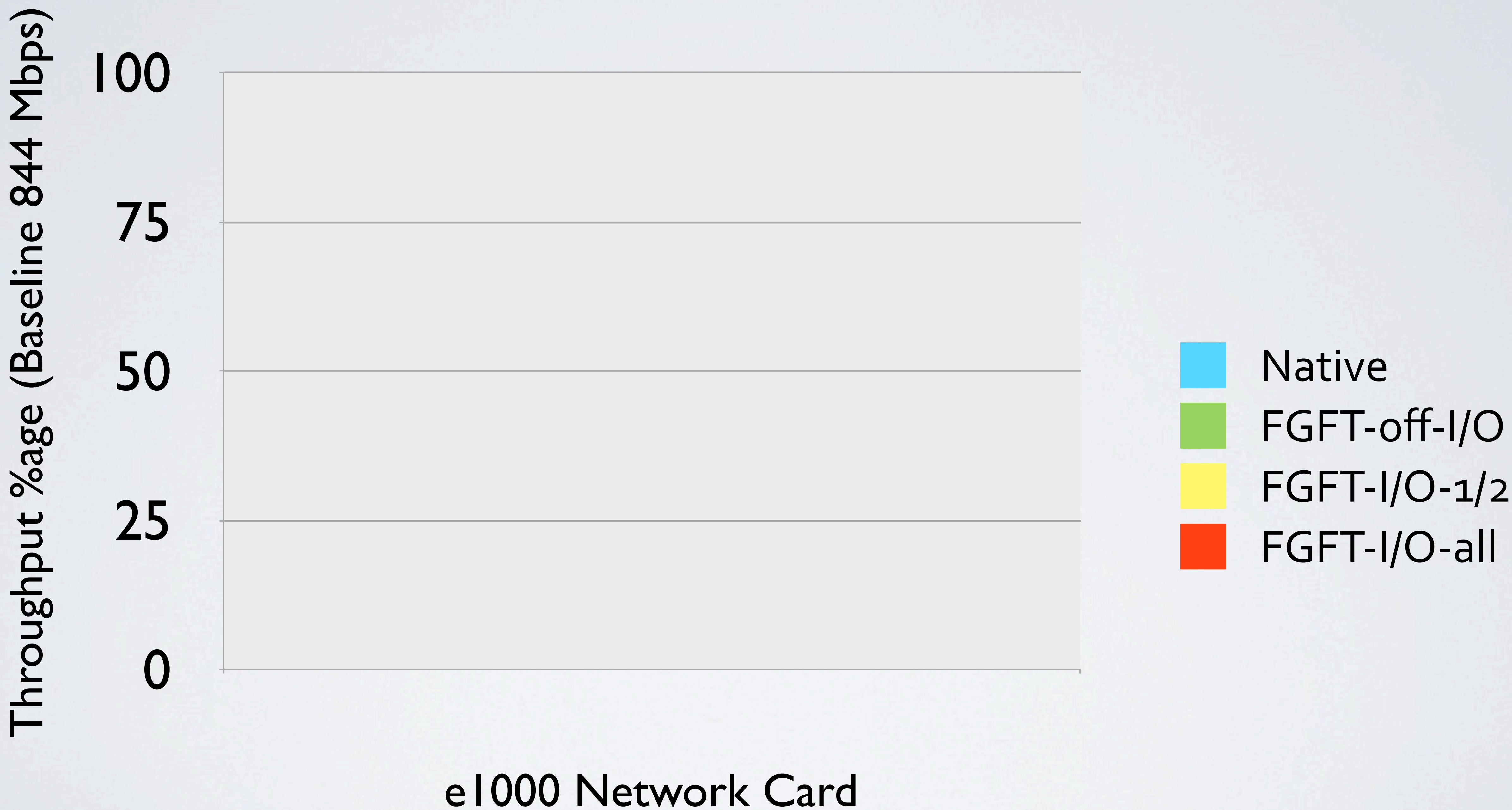
FGFT requires limited programmer effort and needs only 38 lines of new kernel code

Throughput with isolation and recovery

- Native
- FGFT-off-I/O
- FGFT-I/O-1/2
- FGFT-I/O-all

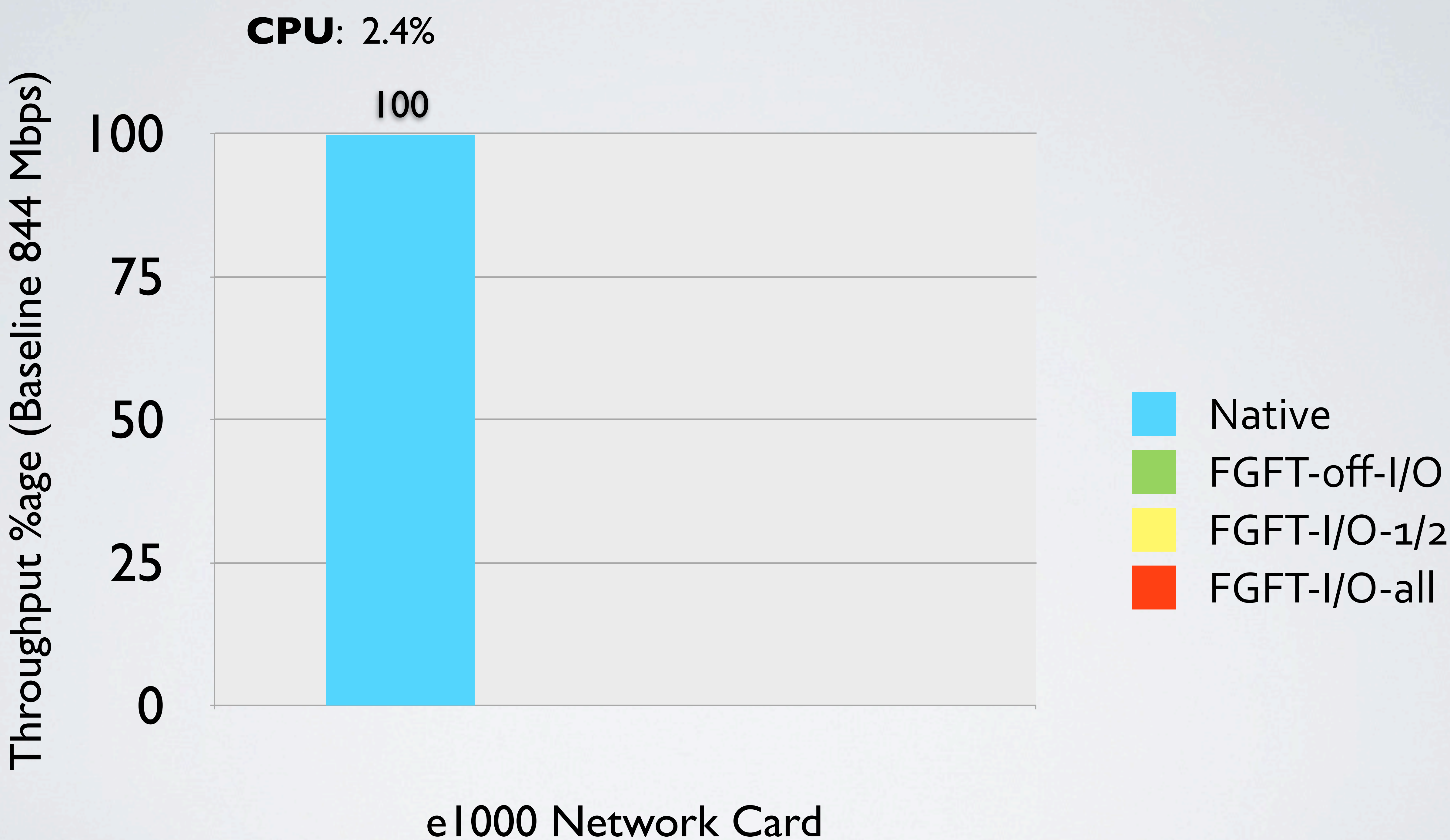
netperf on Intel quad-core machines

Throughput with isolation and recovery



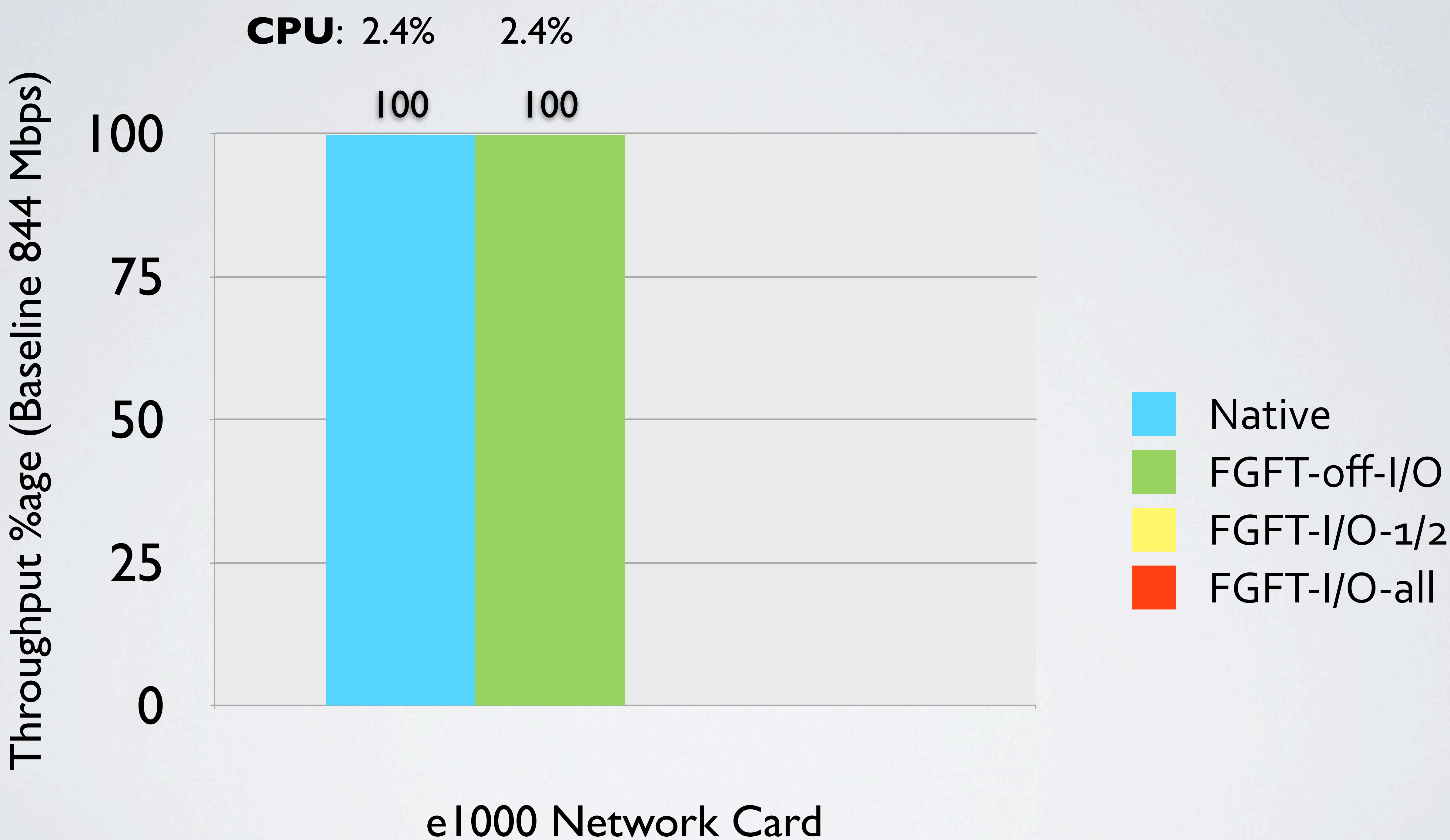
netperf on Intel quad-core machines

Throughput with isolation and recovery



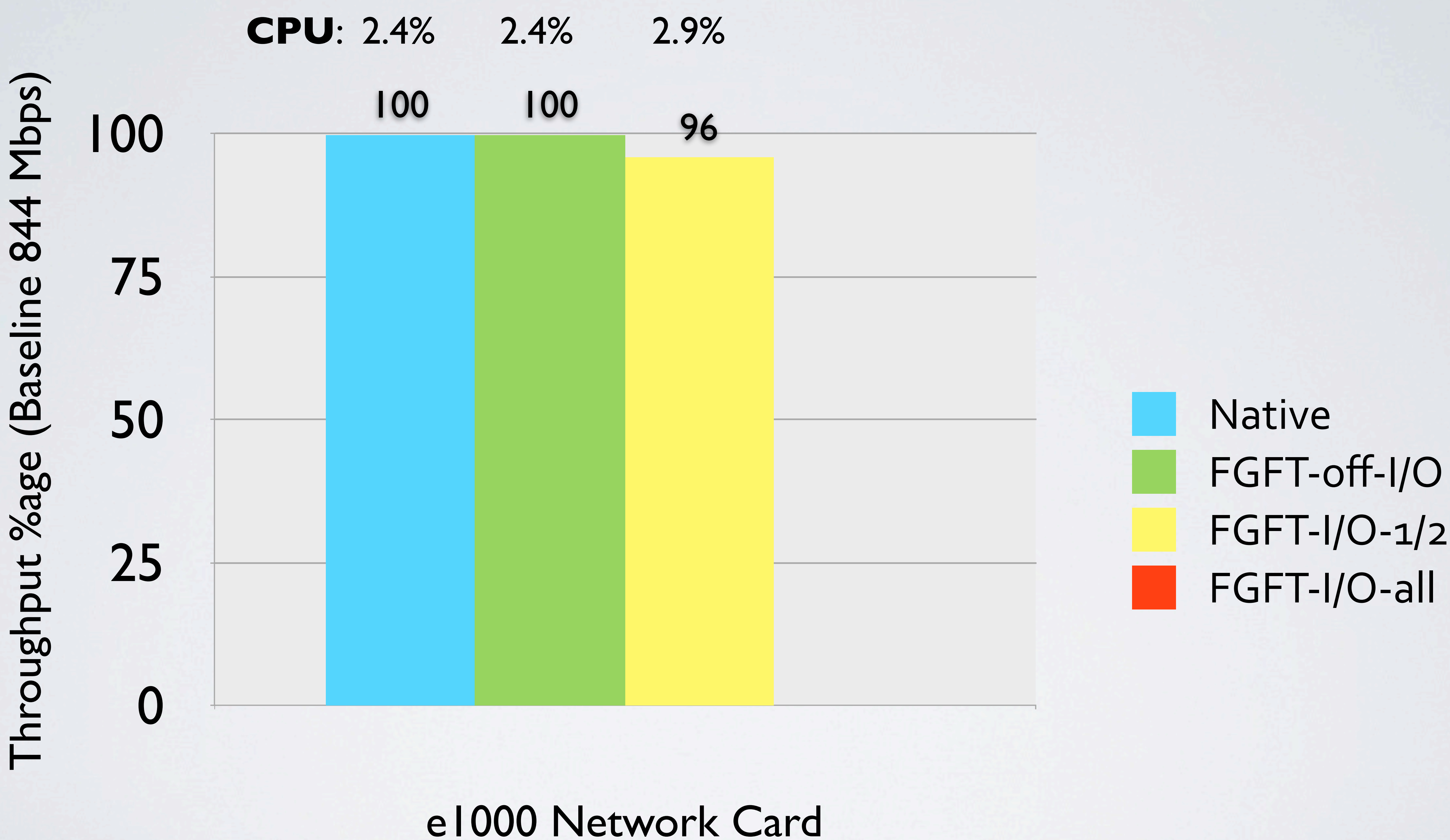
netperf on Intel quad-core machines

Throughput with isolation and recovery



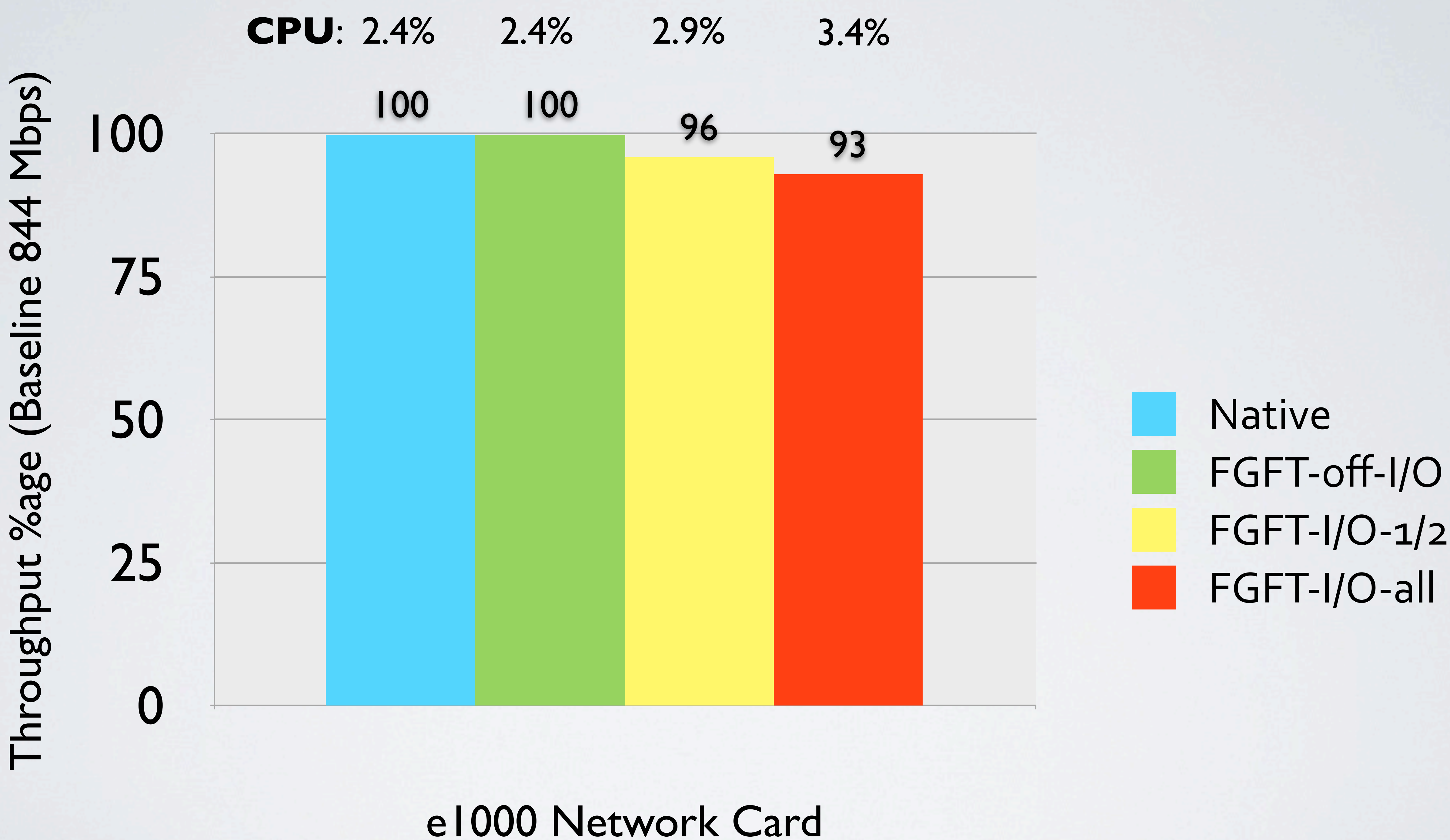
netperf on Intel quad-core machines

Throughput with isolation and recovery



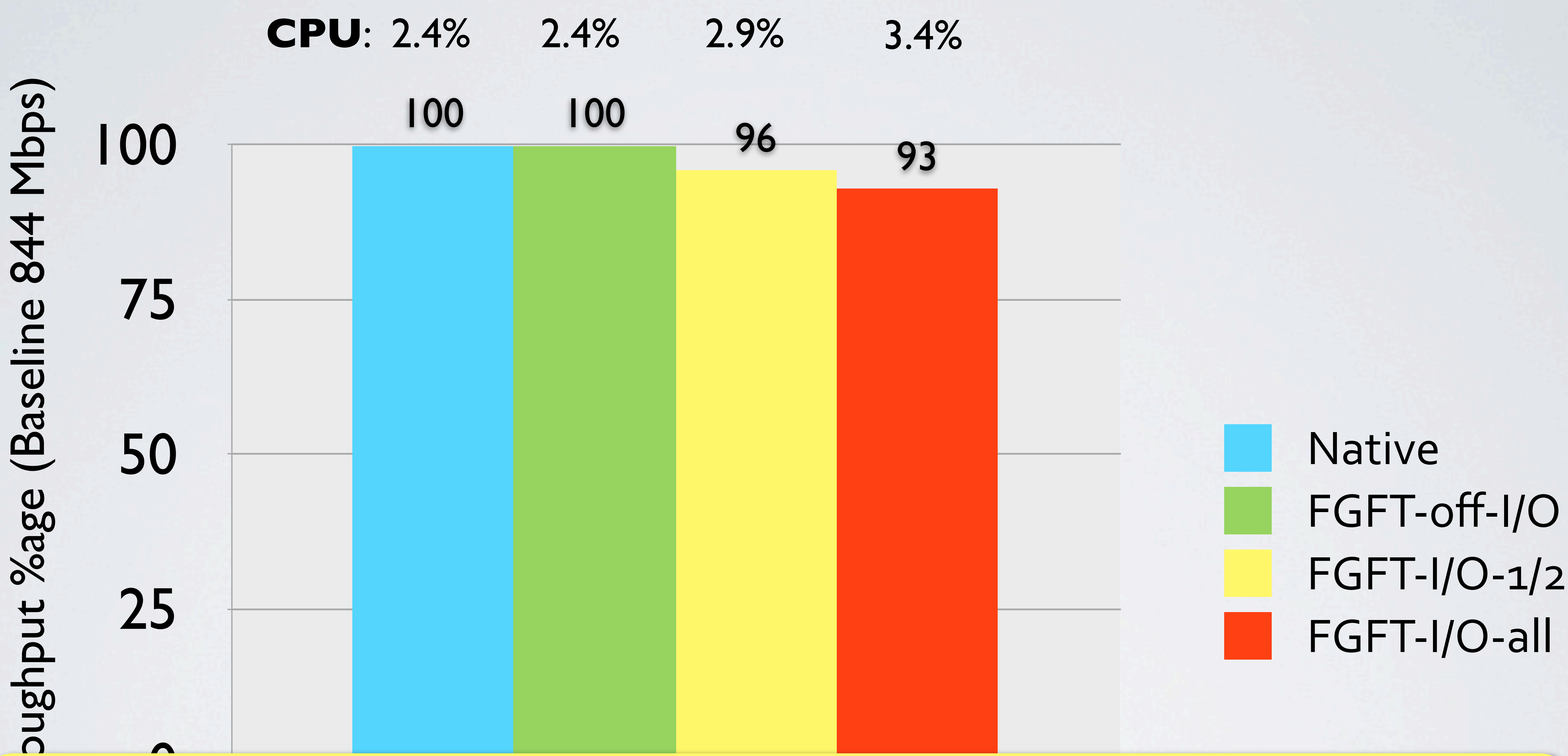
netperf on Intel quad-core machines

Throughput with isolation and recovery



netperf on Intel quad-core machines

Throughput with isolation and recovery



FGFT can isolate and recover high bandwidth devices at low overhead without adding kernel subsystems

netperf on Intel quad-core machines

Summary

Summary

- **Fine-Grained Fault tolerance based on a *pay-as-you go* model**
  - **Provides fault tolerance at incremental performance costs and programmer efforts**
- **Introduces fast checkpointing for drivers**
  - **Device checkpoints average ~20micros**
  - **Reduces recovery time significantly**
  - **Should be explored in other domains apart from fault tolerance like fast reboot, upgrade etc.**

Questions

Asim Kadav

- **http://cs.wisc.edu/~kadav**
- **kadav@cs.wisc.edu**