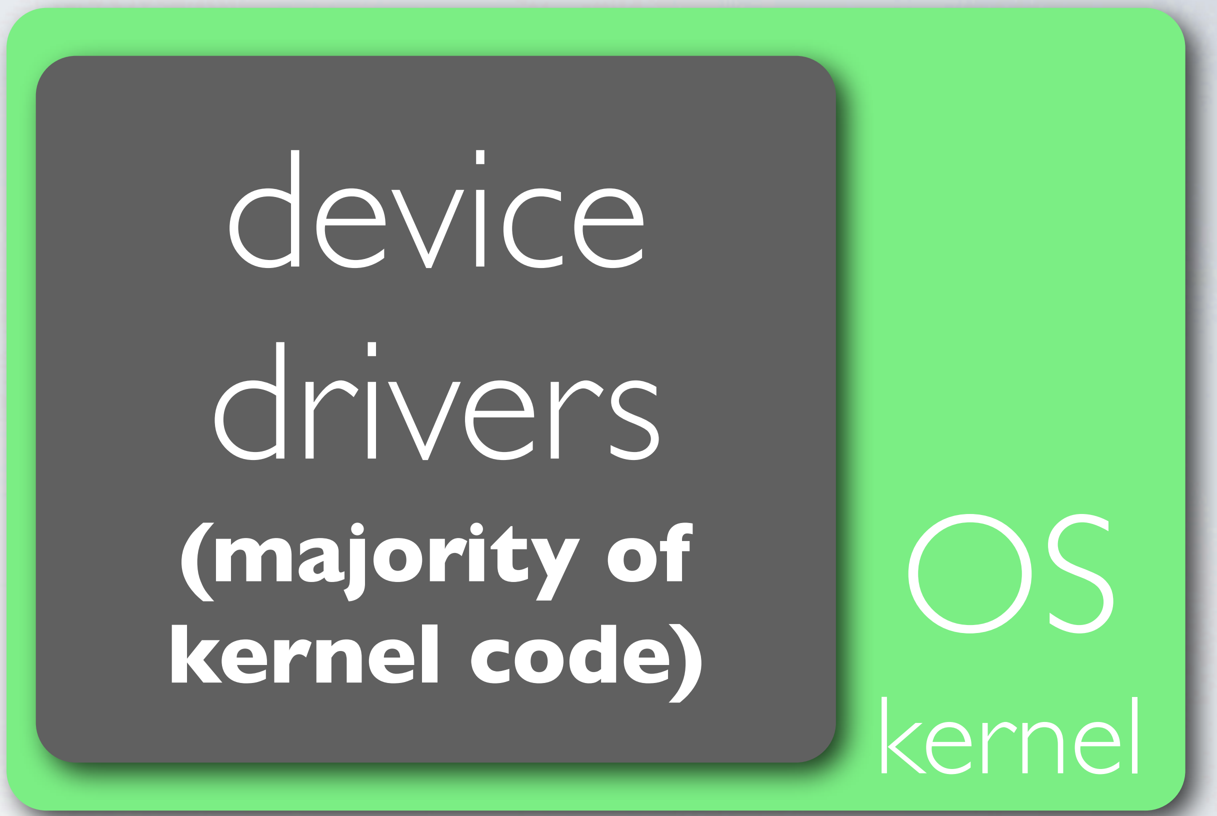
Fine-Grained Fault Tolerance using Device Checkpoints

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

3rd party developers

+

device drivers

(majority of kernel code)

OS kernel

The (old) elephant in the room

3rd party developers

+

device drivers

(majority of kernel code)

OS kernel

Recipe for disaster

Extensive past work on reliability research

Improvement	System	Validation		
		Drivers	Bus	Classes
Isolation	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
Recovery	Shadow Drivers [OSDI 04]	13	1	3
Static analysis tools	Windows SDV [Eurosys 06]	All	All	All
	Coverity [CACM 10]	All	All	All
	Cocinelle [Eurosys 08]	All	All	All

Extensive past work on reliability research

Improvement	System	Validation		
		Drivers	Bus	Classes
Isolation	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
Recovery	Shadow Drivers [OSDI 04]	13	1	3
Static analysis tools	Windows SDV [Eurosys 06]	All	All	All
	Coverity [CACM 10]	All	All	All
	Cocinelle [Eurosys 08]	All	All	All

Extensive past work on reliability research

Improvement	System	Validation		
		Drivers	Bus	Classes
Isolation	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
Recovery	Shadow Drivers [OSDI 04]	13	1	3
Static analysis tools	Windows SDV [Eurosys 06]	All	All	All
	Coverity [CACM 10]	All	All	All
	Cocinelle [Eurosys 08]	All	All	All

Extensive past work on reliability research

Improvement	System	Validation		
		Drivers	Bus	Classes
Isolation	Nooks [SOSP 03]	6	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
Recovery	Shadow Drivers [OSDI 04]	13	1	3
Static analysis tools	Windows SDV [Eurosys 06]	All	All	All
	Coverity [CACM 10]	All	All	All
	Cocinelle [Eurosys 08]	All	All	All

Observation 1: Solutions that limit changes to kernel and apply to lots of drivers have real impact

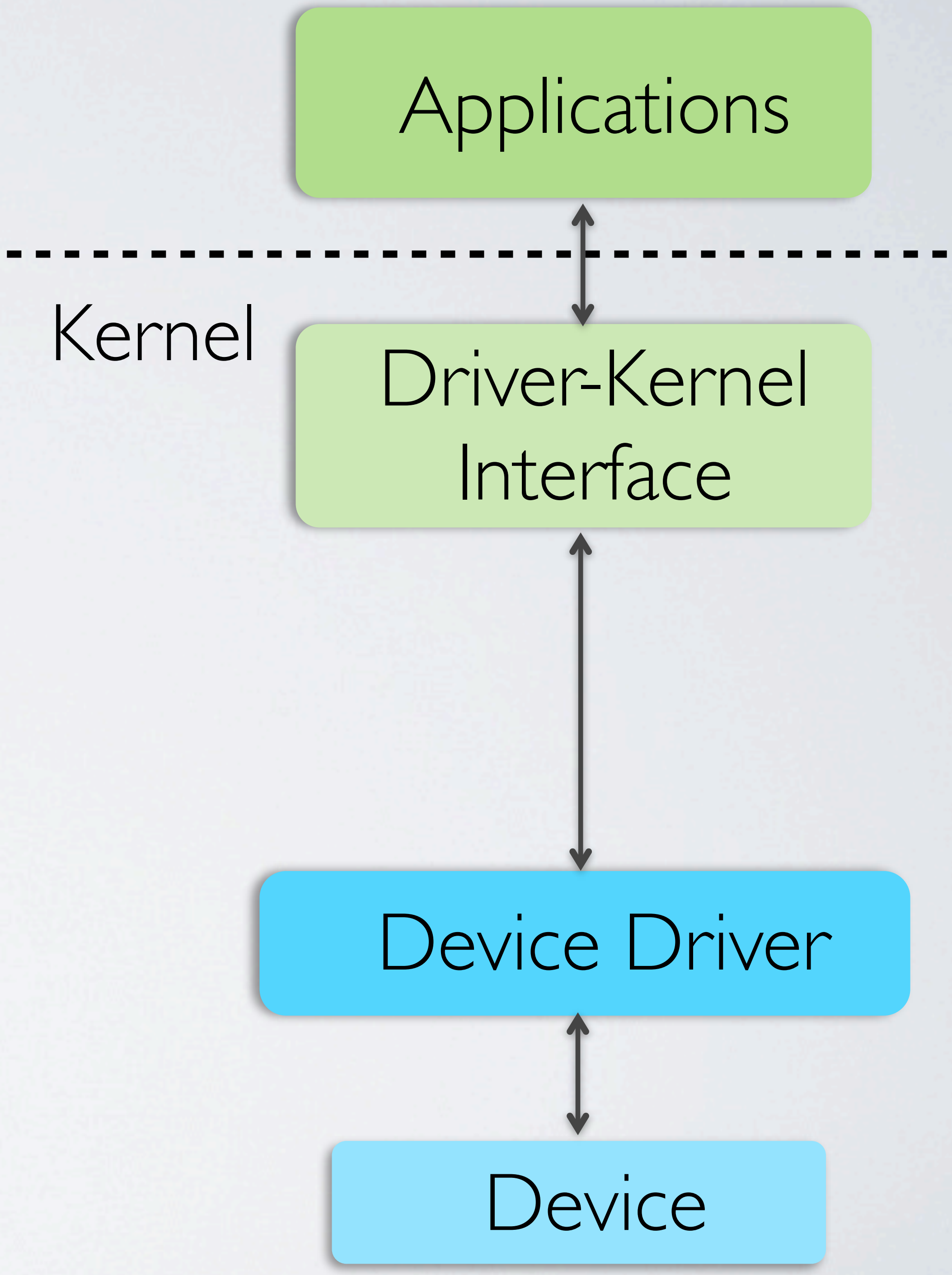
Extensive past work on reliability research

Improvement	System	Validation		
		Drivers	Bus	Classes
Isolation	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
Recovery	Shadow Drivers [OSDI 04]	13	1	3
Static analysis tools	Windows SDV [Eurosys 06]	All	All	All
	Coverity [CACM 10]	All	All	All

Observation 2: Most systems focus on improving isolation and detection and not on recovery

Driver failure recovery limited to driver restart

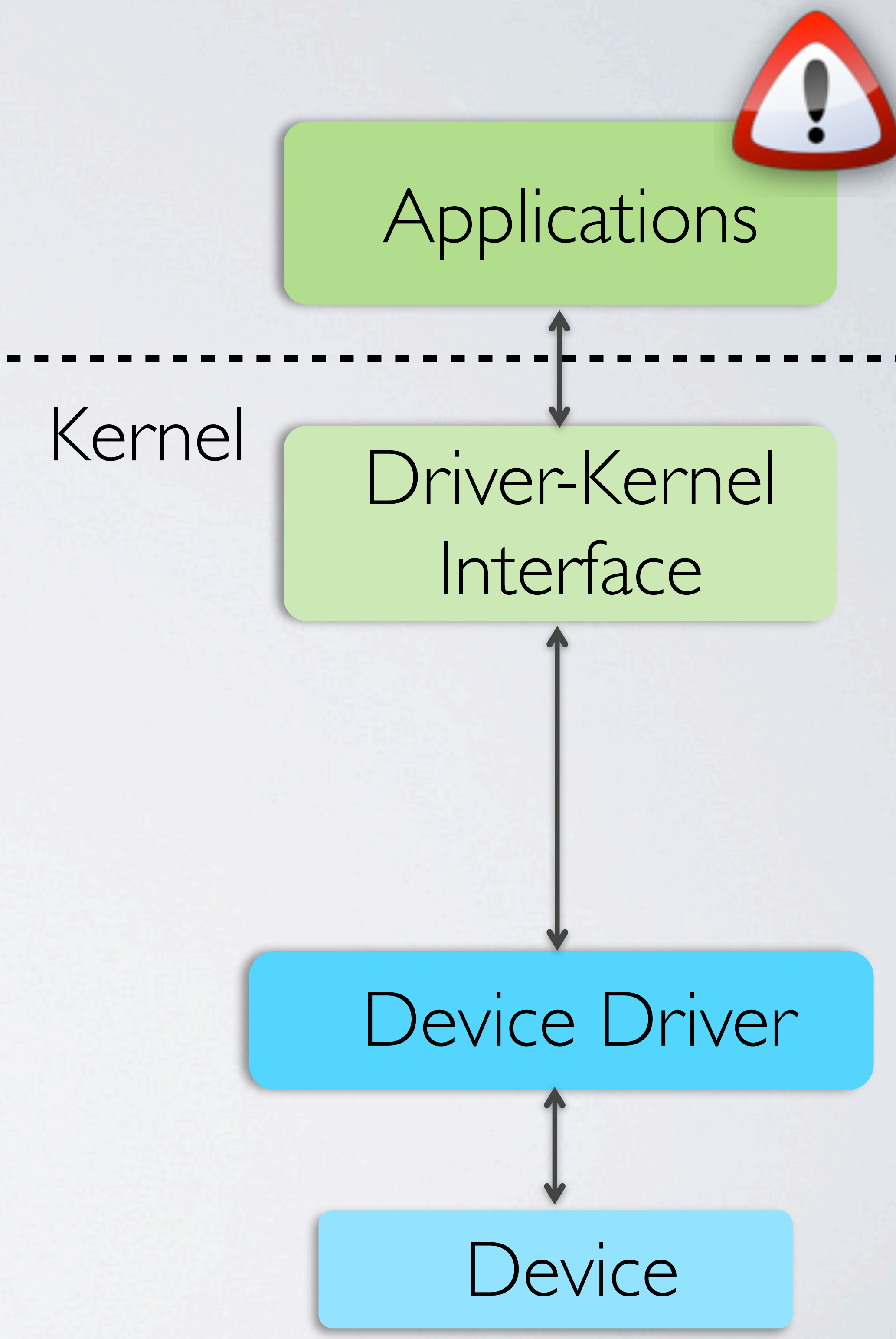
- Restart driver upon failure
  - Safedrive and MINIX approach
  - Can break applications



Shadow drivers

Driver failure recovery limited to driver restart

- Restart driver upon failure
  - Safedrive and MINIX approach
  - Can break applications

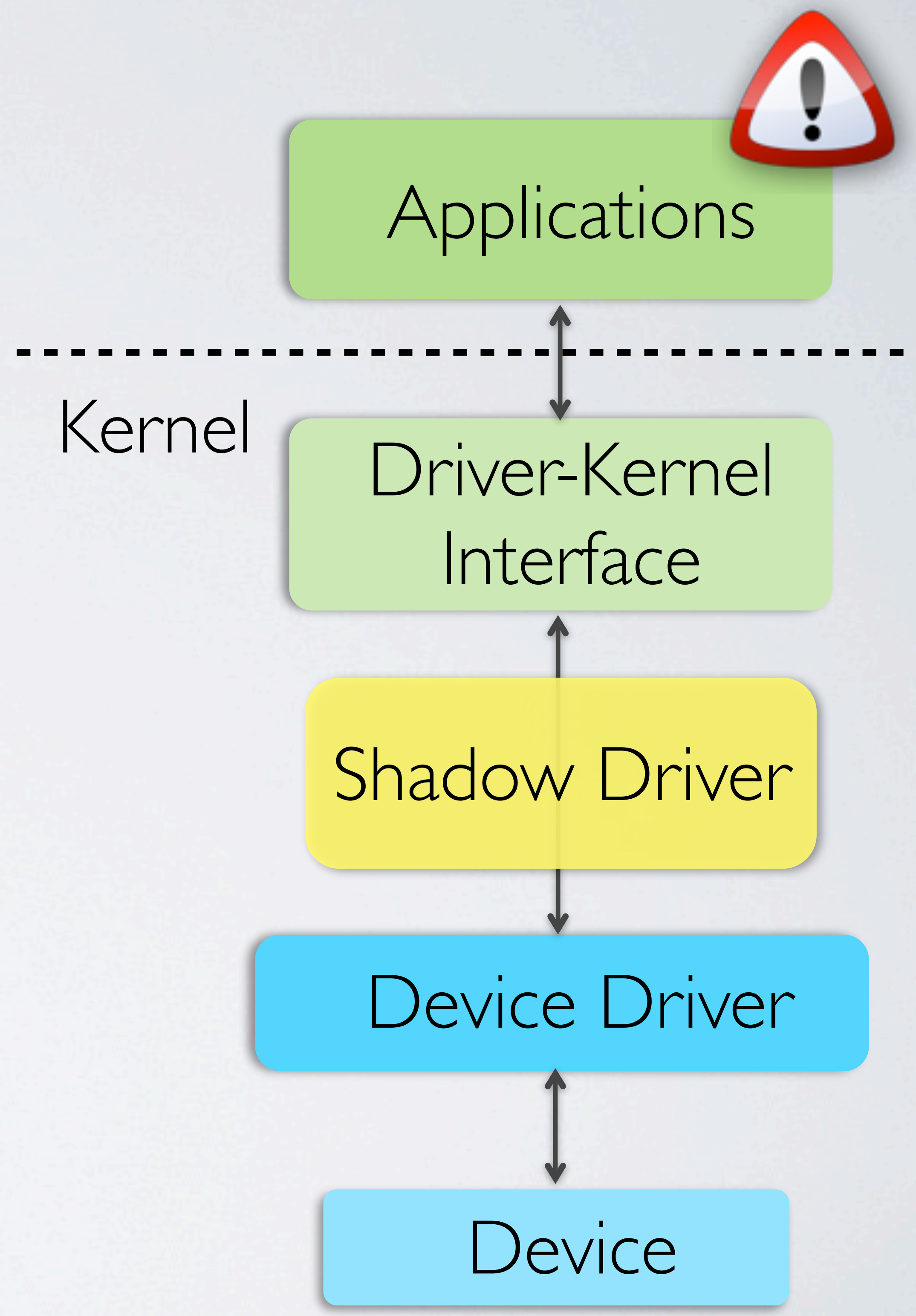


Shadow drivers

Driver failure recovery limited to driver restart

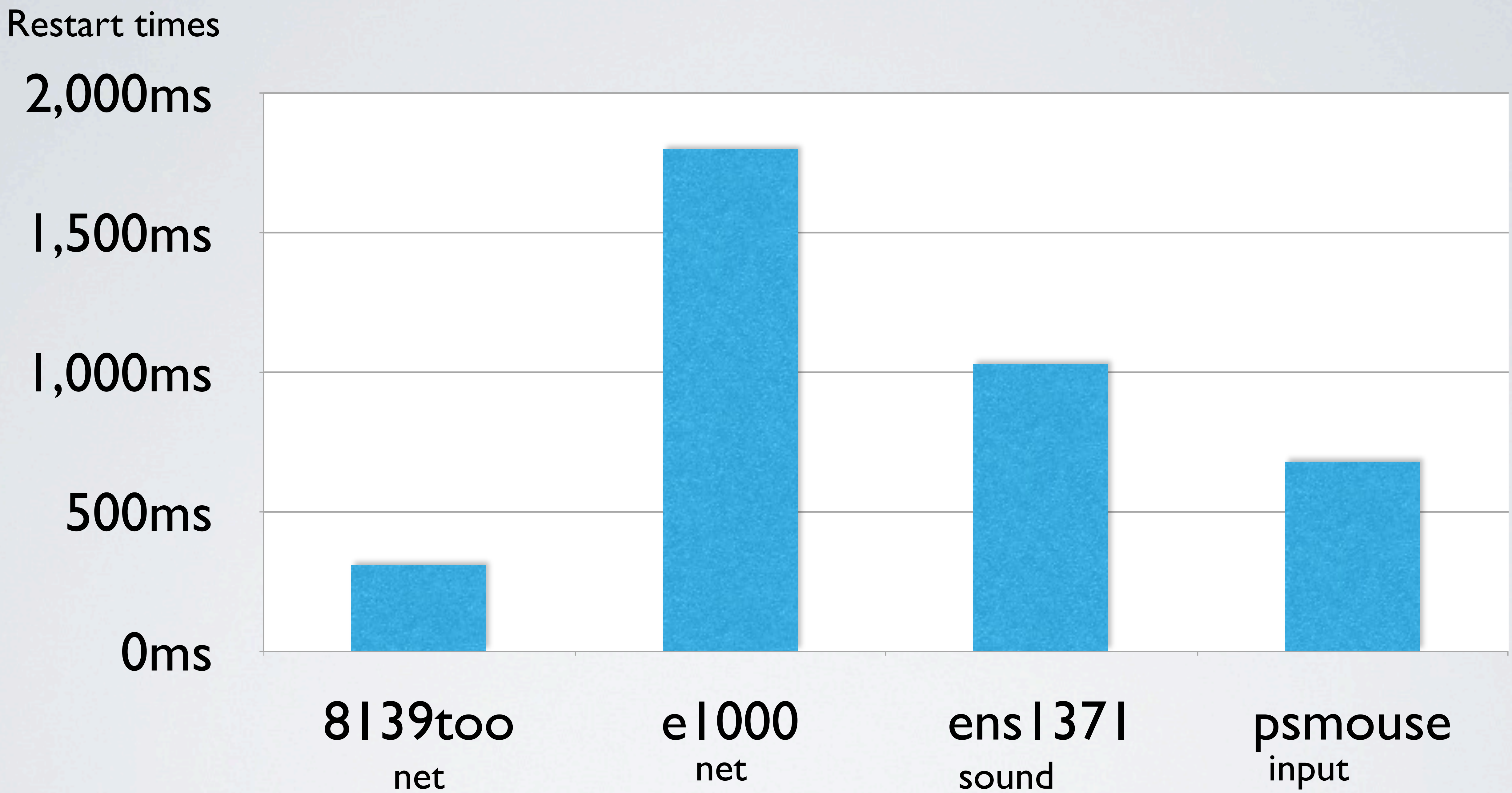
- Restart driver upon failure
  - Safedrive and MINIX approach
  - Can break applications

- Restart and replay upon failure
  - Shadow driver approach
  - Always record state of driver
  - Perform restart and log replay upon failure
  - Transparent to applications



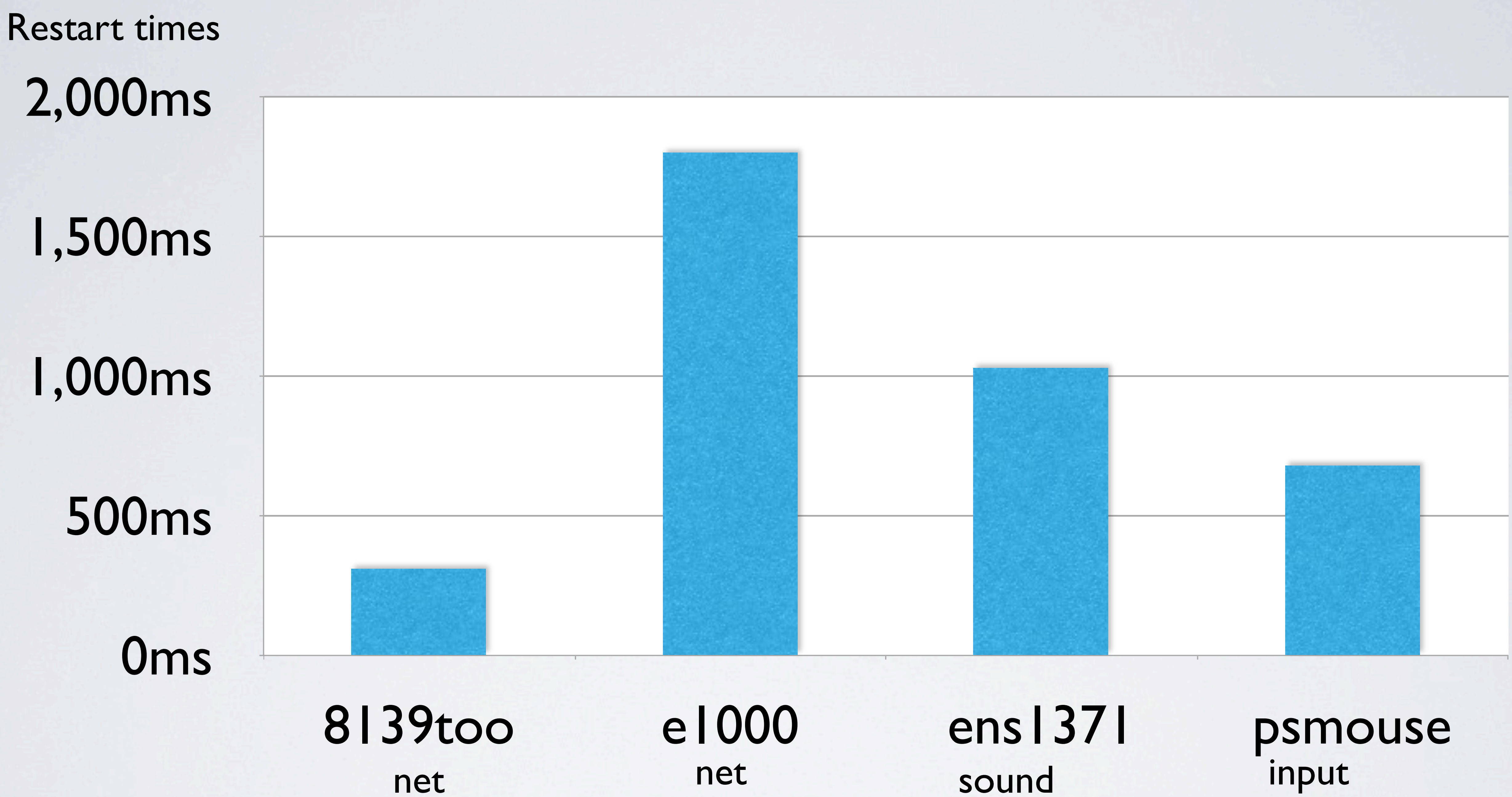
Shadow drivers

Problem 1: Restart based driver recovery is slow



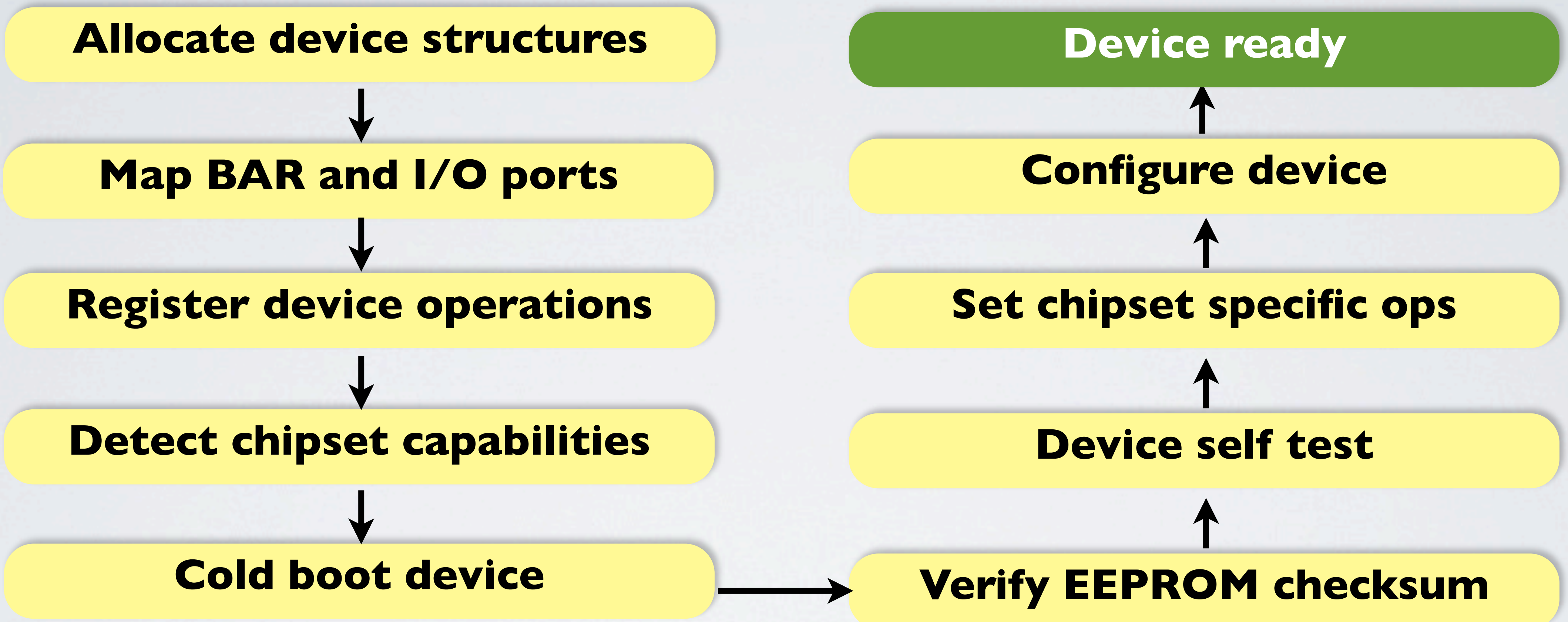
5

Problem 1: Restart based driver recovery is slow

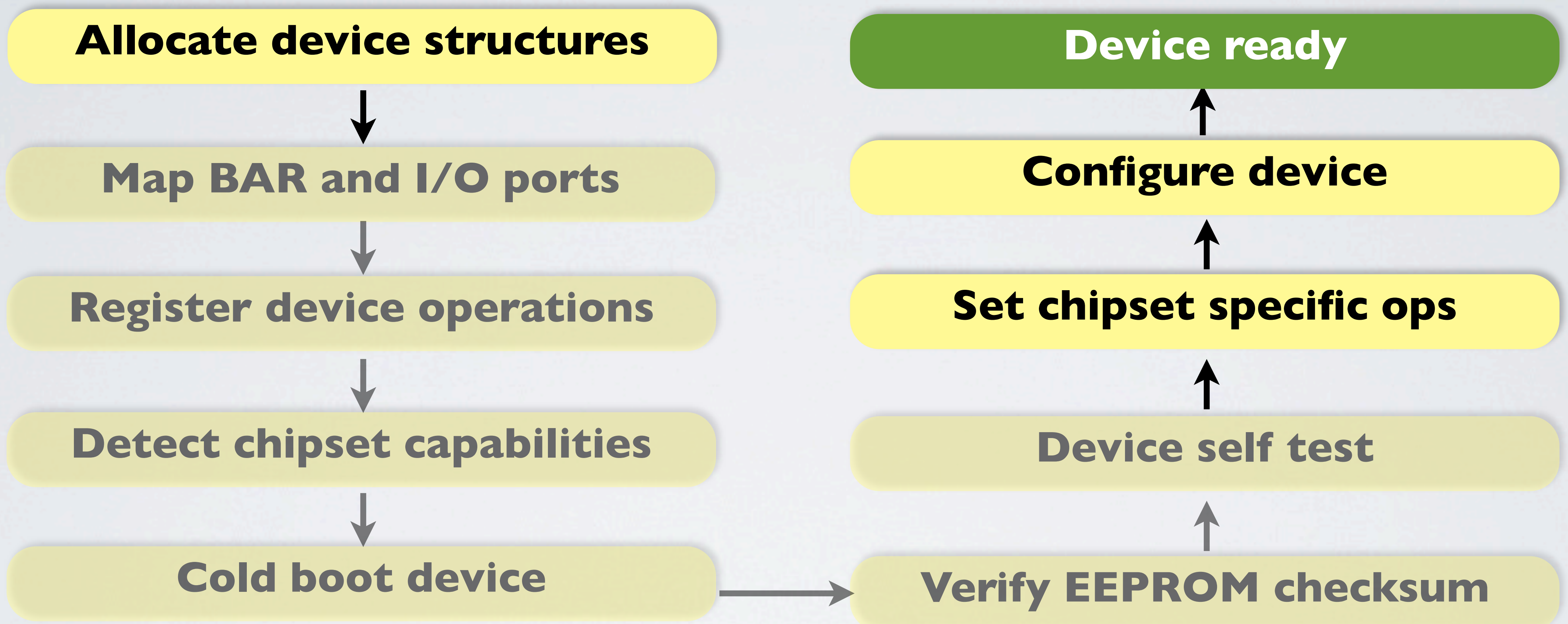


Shadow drivers restart the driver upon failure which can be slow

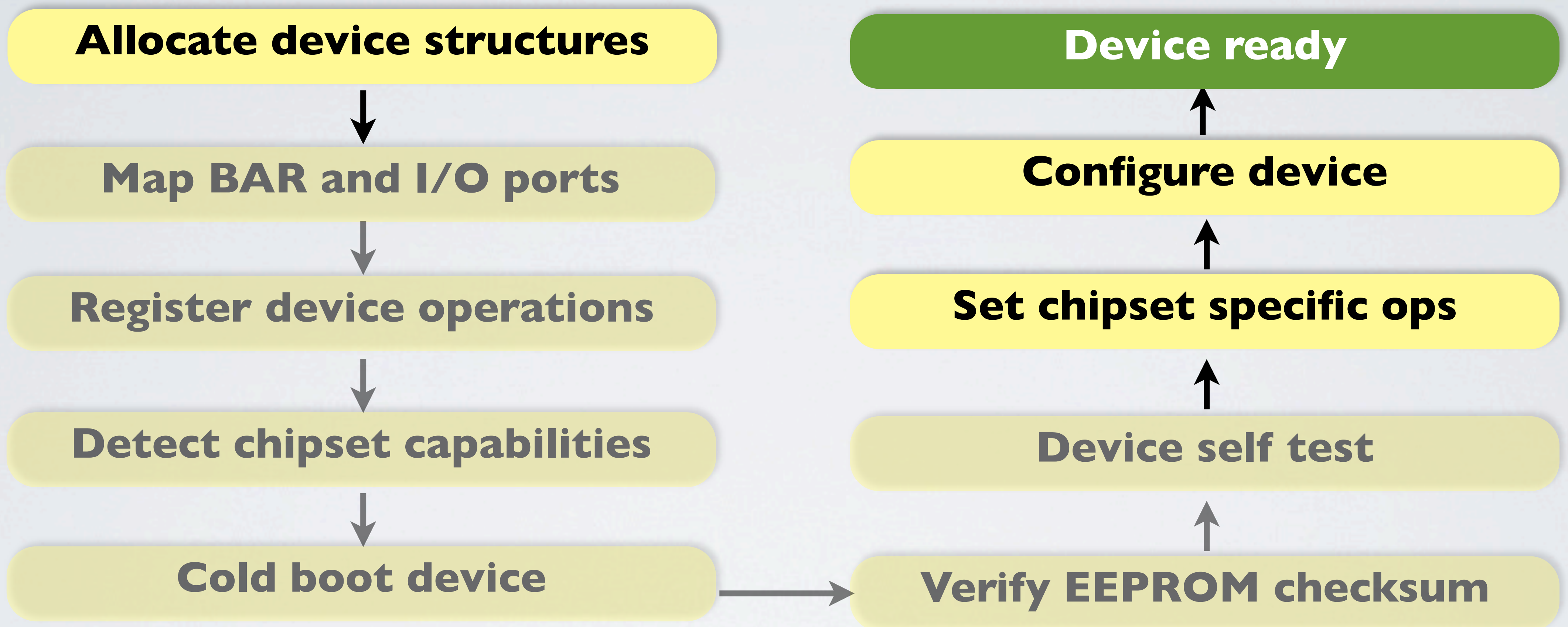
Driver re-initialization probes hardware again



Driver re-initialization probes hardware again

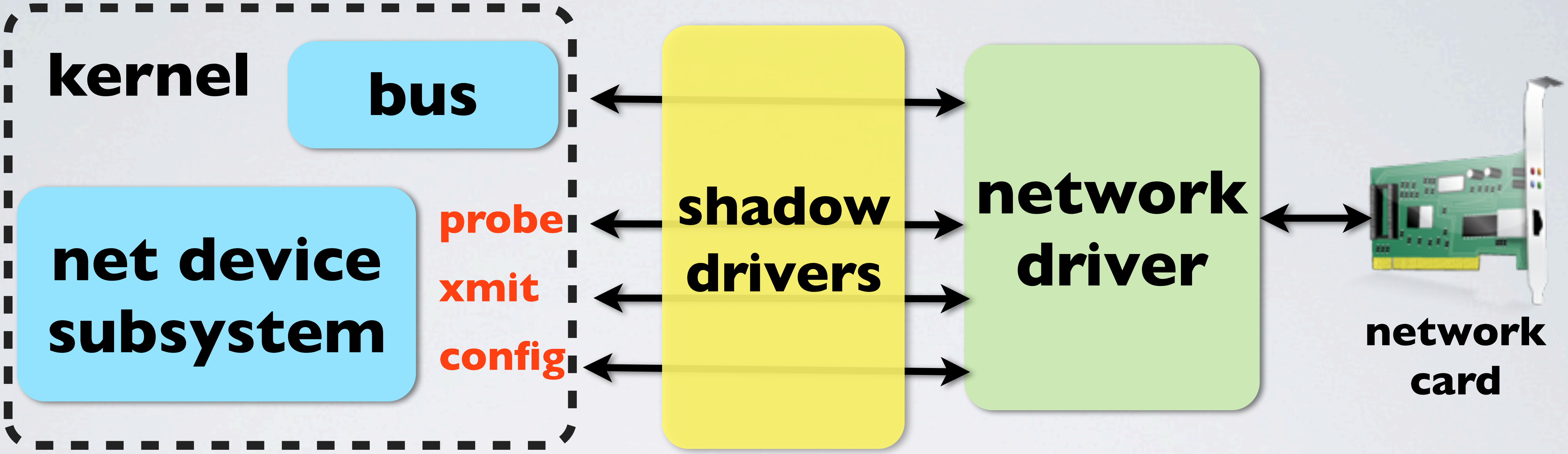


Driver re-initialization probes hardware again



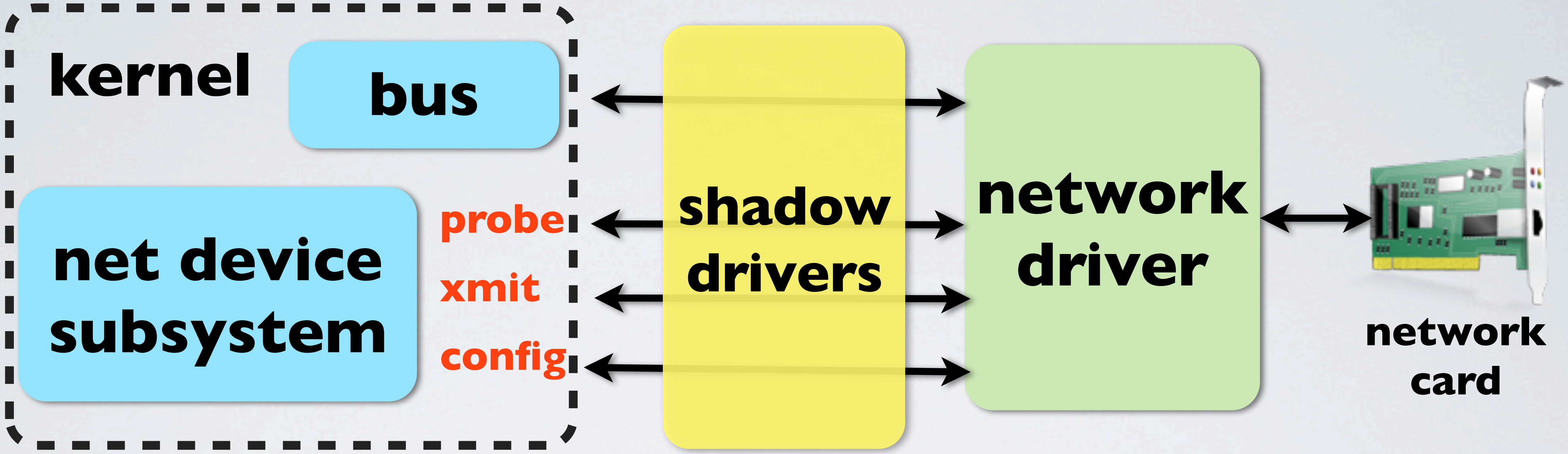
- What does slow device re-initialization hurt?
  - Fault tolerance: Driver recovery
  - Virtualization: Live migration
  - OS functions: Fast reboot

Problem 2: Shadow drivers assume drivers follow class behavior



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

Problem 2: Shadow drivers assume drivers follow class behavior

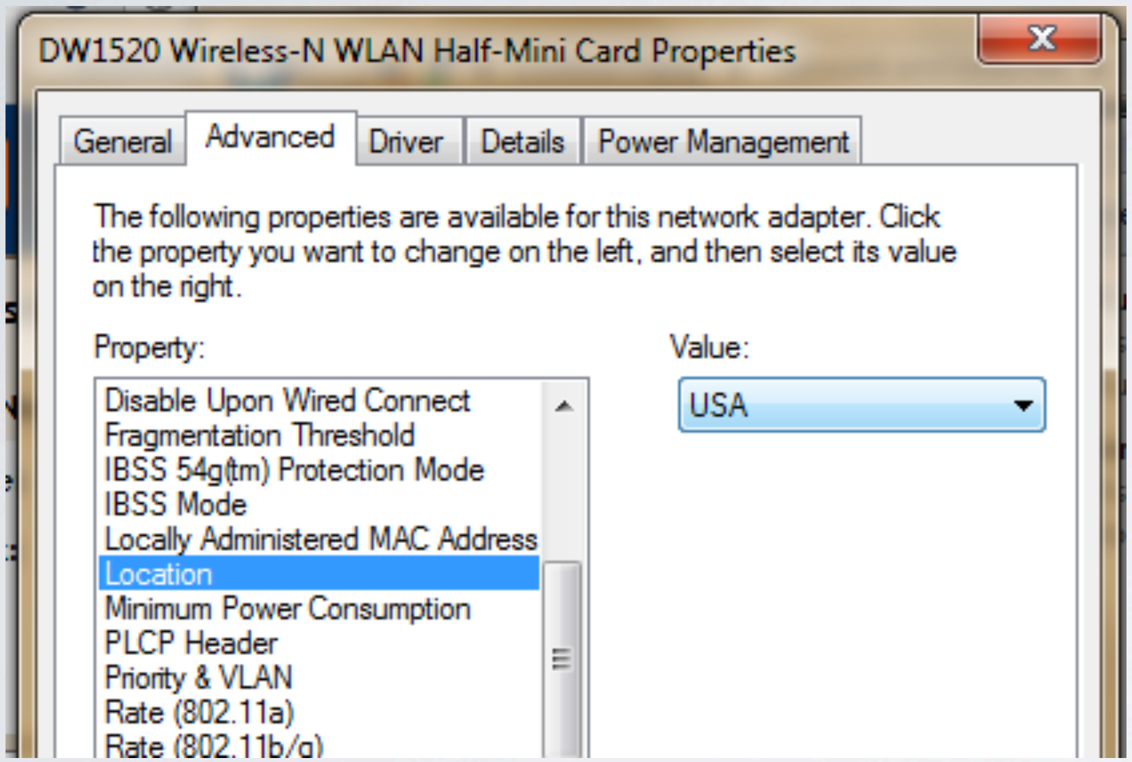


- Class definition includes:
  - Callbacks registered with the bus, device and kernel subsystem

How many drivers follow class behavior and how much code does this add and

Problem 2(a): Drivers do behave outside class definitions

- **Non-class behavior that affects recovery:**
  - **procfs/sysfs interactions and unique ioctls**



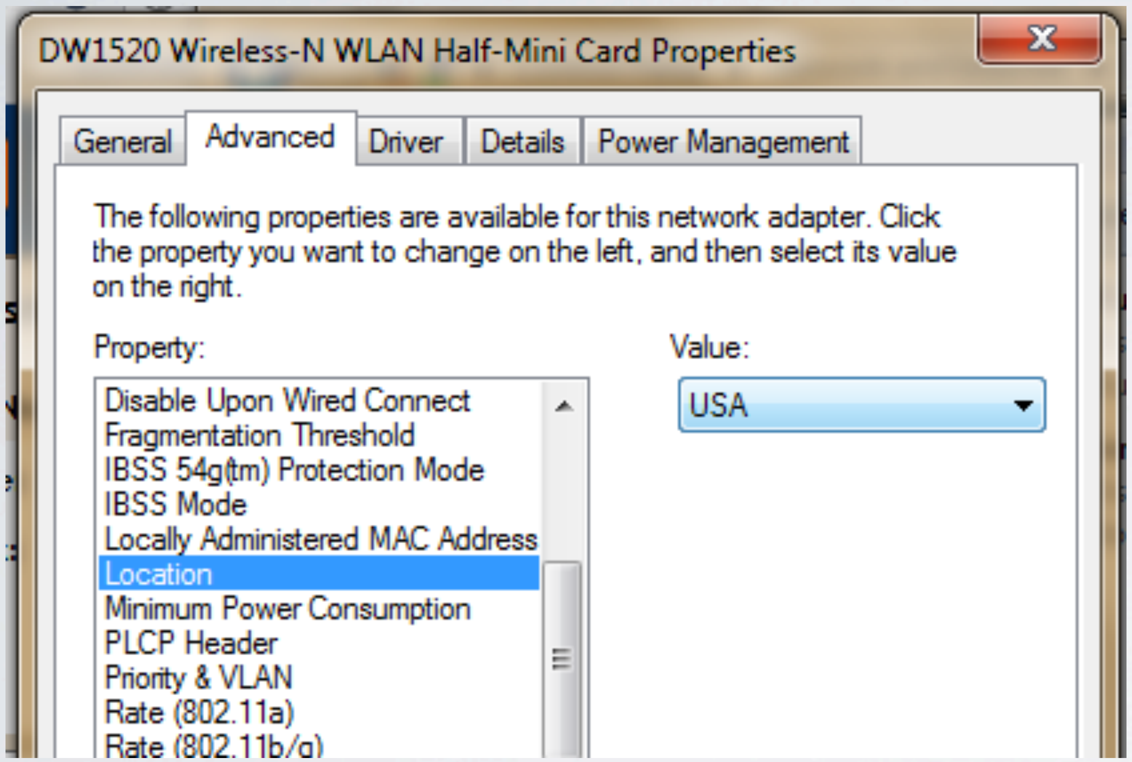
Windows WLAN card config via private ioctls

```
$ echo 1 > /sys/class/sound/mixer/
  device/enable
```

Linux sound card config via sysfs

Problem 2(a): Drivers do behave outside class definitions

- **Non-class behavior that affects recovery:**
  - **procfs/sysfs interactions and unique ioctls**



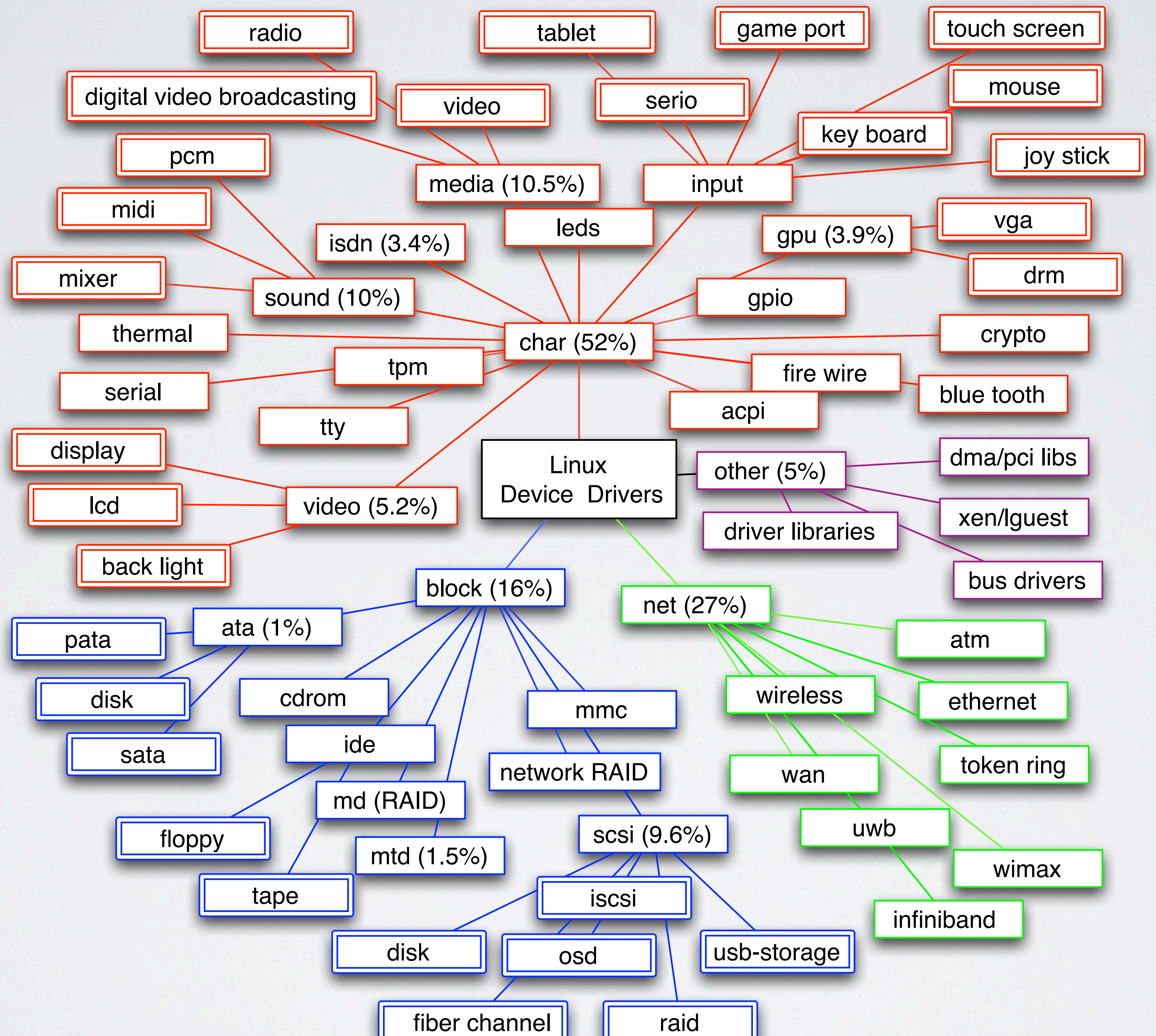
Windows WLAN card config via private ioctls

```
$ echo 1 > /sys/class/sound/mixer/
    device/enable
```

Linux sound card config via sysfs

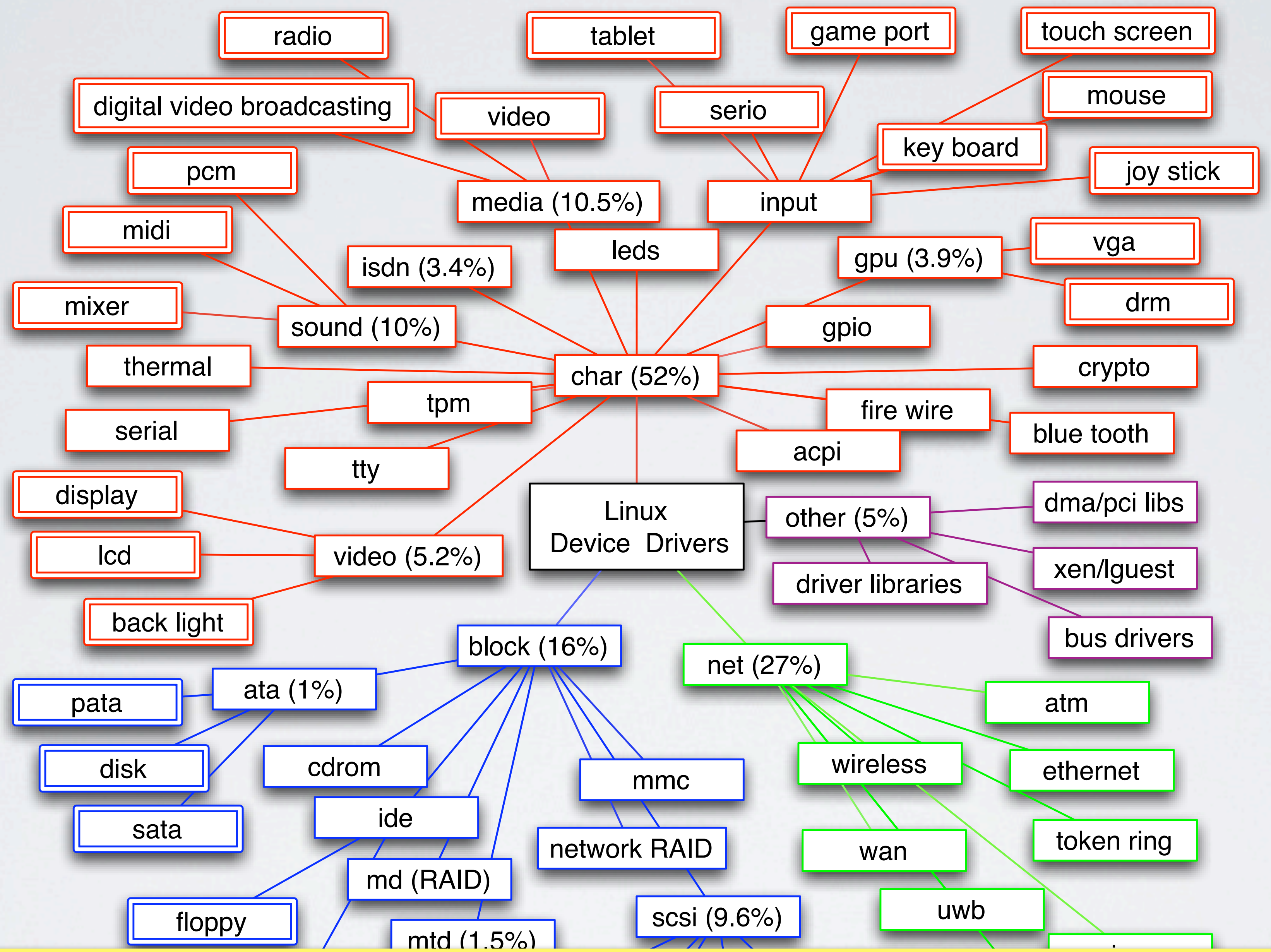
At least 16% of drivers have non-class behavior and may not recover correctly using shadow drivers

Problem 2(b): Too many classes



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

Problem 2(b): Too many classes



Class-specific driver recovery leads to a large kernel recovery subsystem

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

Fine-Grained Fault Tolerance (FGFT)

10

Fine-Grained Fault Tolerance (FGFT)

Fine-grained Isolation

- Runs driver entry points like transactions
- Relies on code generation to limit new code in kernel

Fine-Grained Fault Tolerance (FGFT)

Fine-grained Isolation

- ⋆ Runs driver entry points like transactions
- ⋆ Relies on code generation to limit new code in kernel

Checkpoint-based recovery

- ⋆ Provides fast and correct recovery semantics

Fine-Grained Fault Tolerance (FGFT)

Fine-grained Isolation

- ★ Runs driver entry points like transactions
- ★ Relies on code generation to limit new code in kernel

Checkpoint-based recovery

- ★ Provides fast and correct recovery semantics

- ★ Requires incremental overhead/changes to drivers
- ★ Shifts burden of fault tolerance to faulty code

Outline

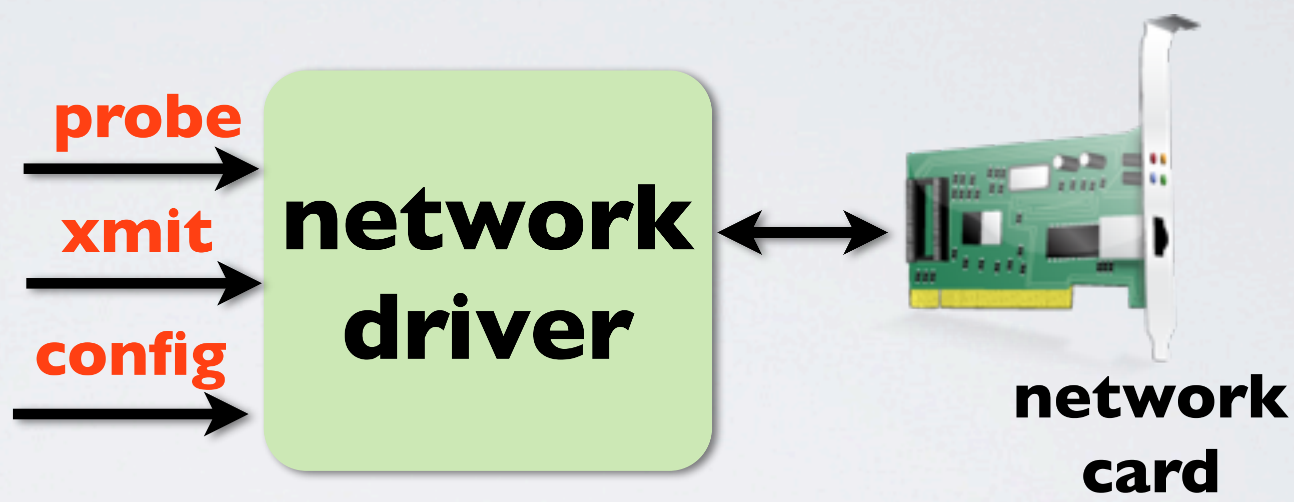
Introduction

Fine-grained isolation

Checkpoint-based recovery

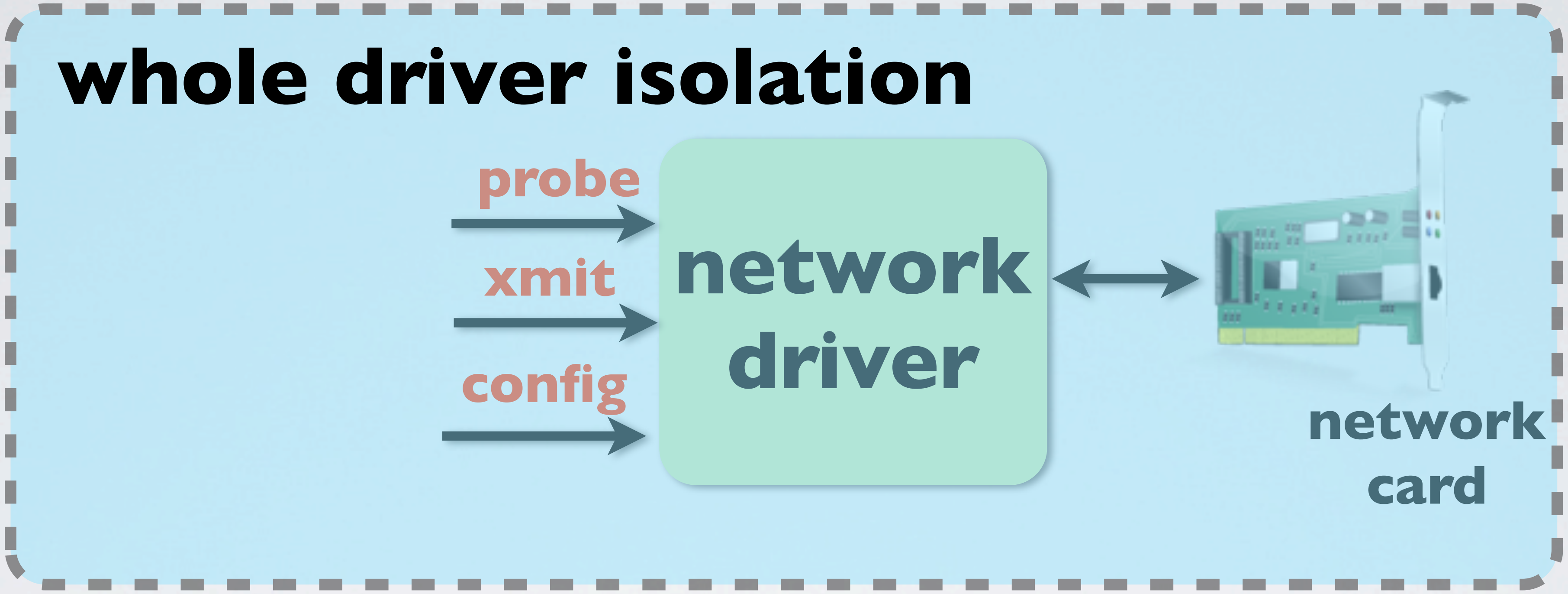
Evaluation and Conclusions

Unit of fault tolerance: Driver entry point

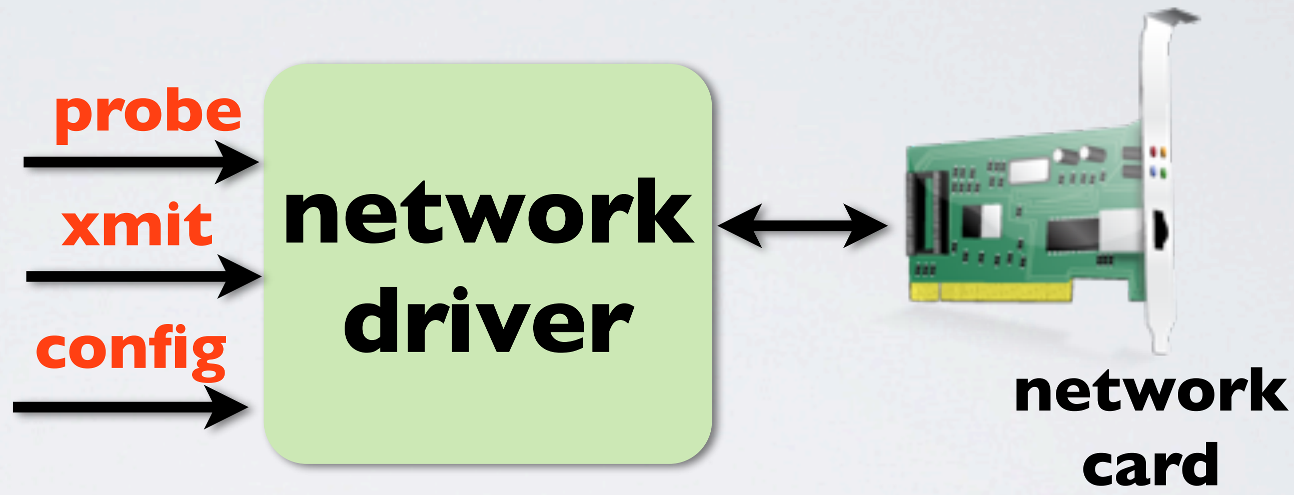


12

Unit of fault tolerance: Driver entry point

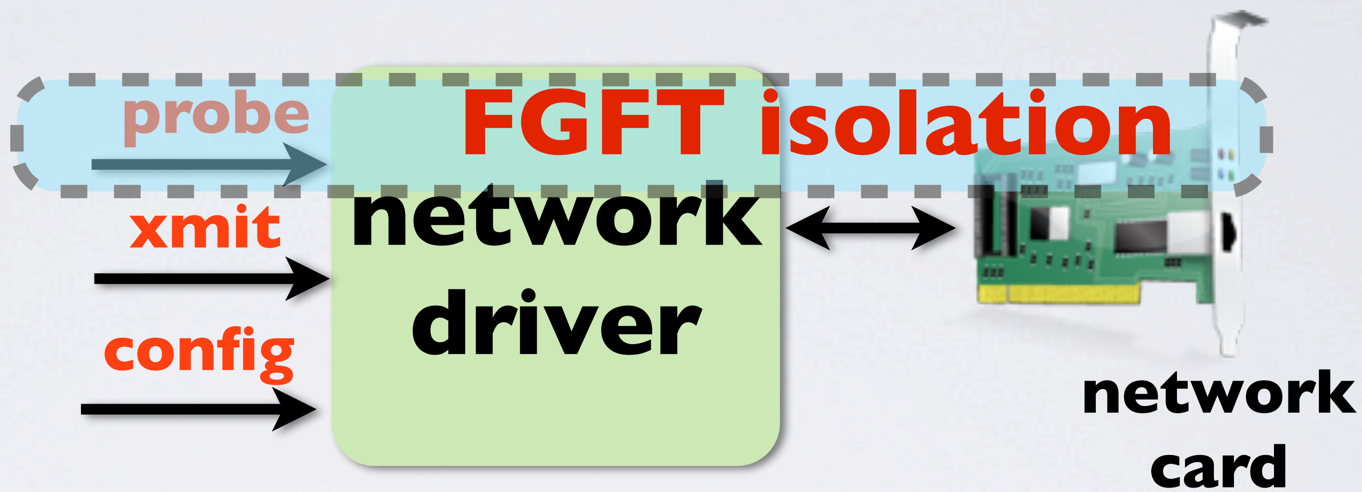


Unit of fault tolerance: Driver entry point

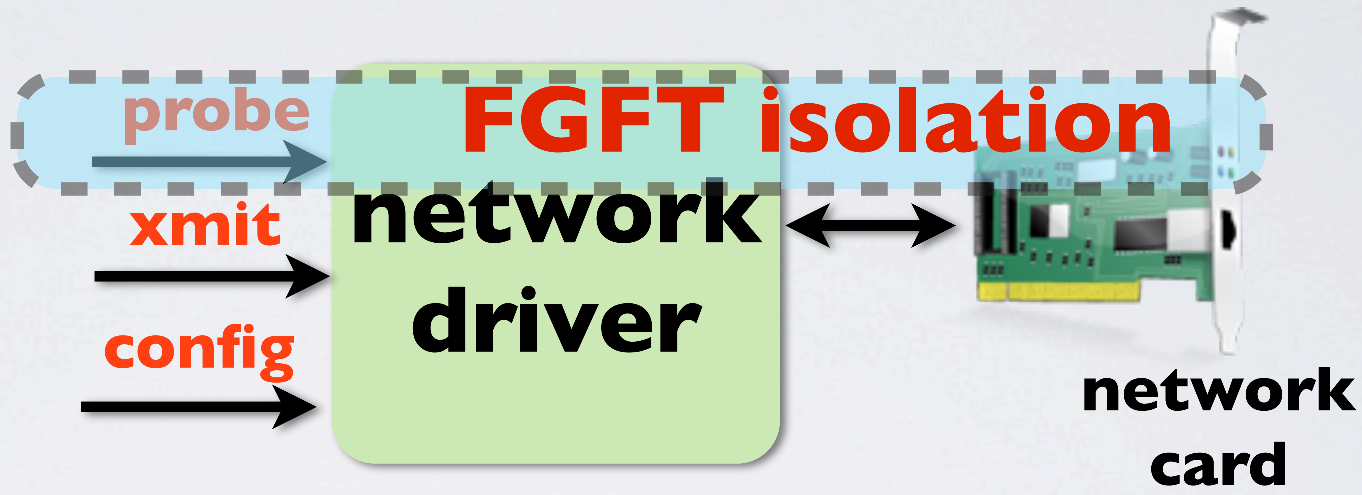


12

Unit of fault tolerance: Driver entry point

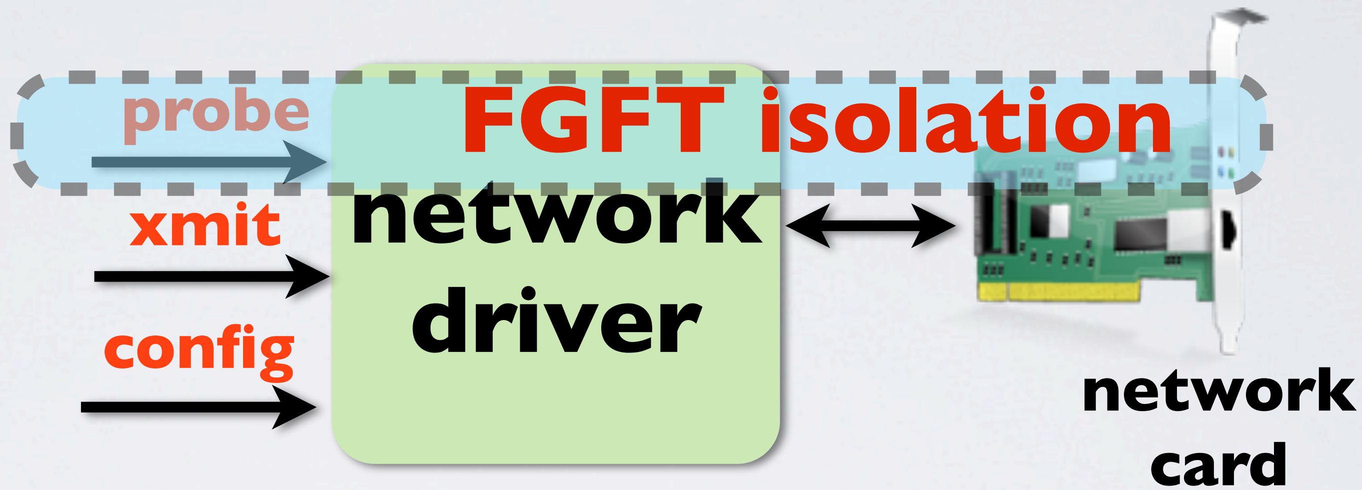


Unit of fault tolerance: Driver entry point



★ **Provide fault tolerance to specific driver entry points**

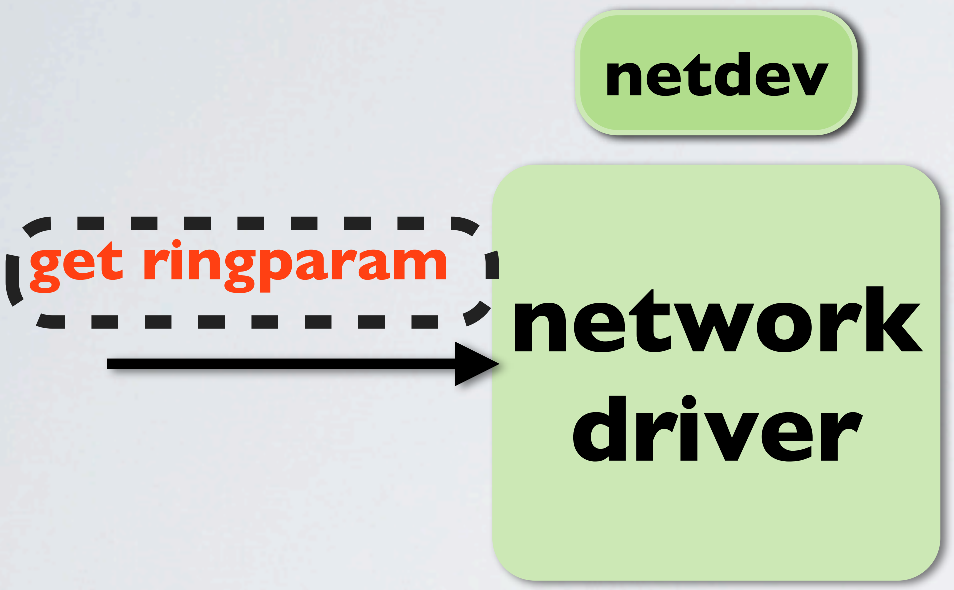
Unit of fault tolerance: Driver entry point



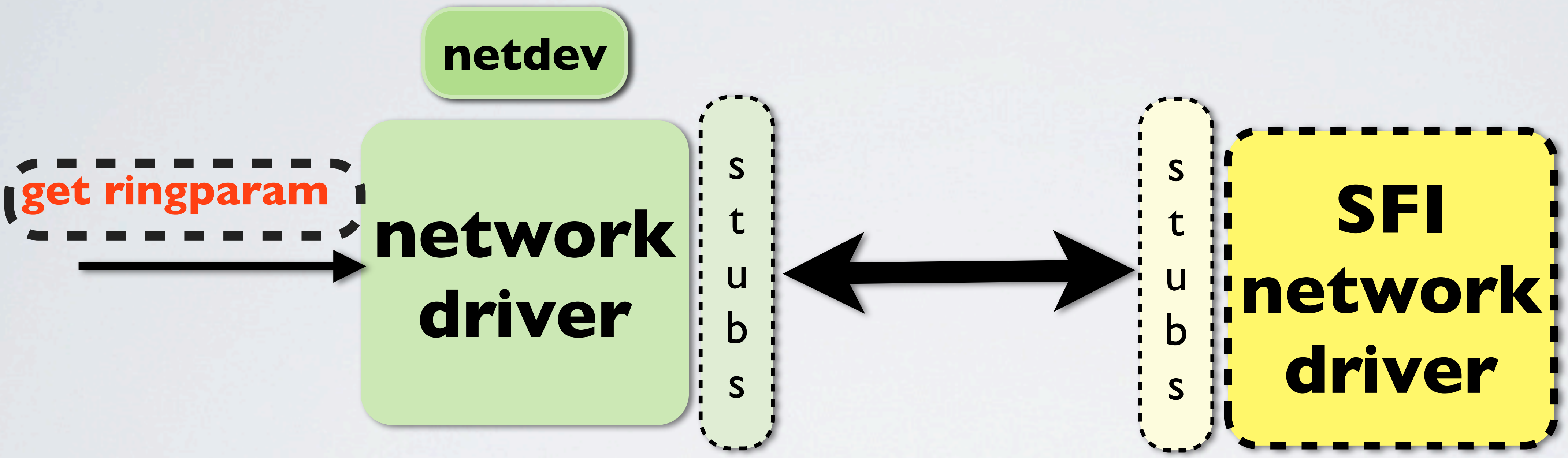
★ **Provide fault tolerance to specific driver entry points**

★ **Can be applied to untested code or code marked suspicious by static or runtime tools**

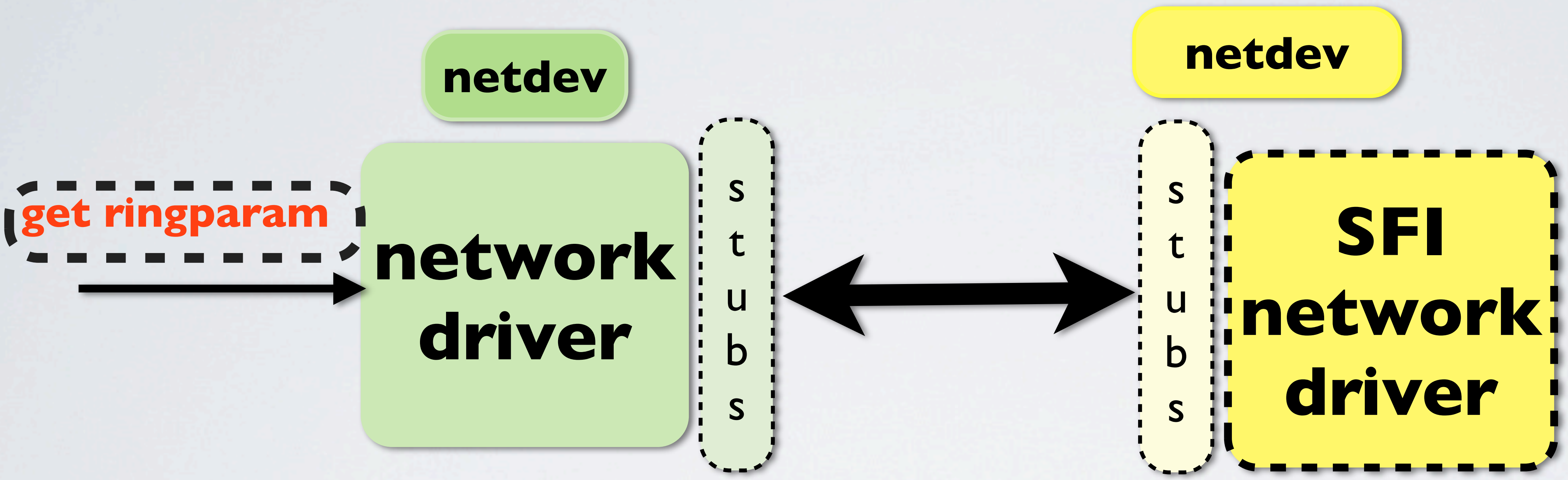
Transactional support through code generation



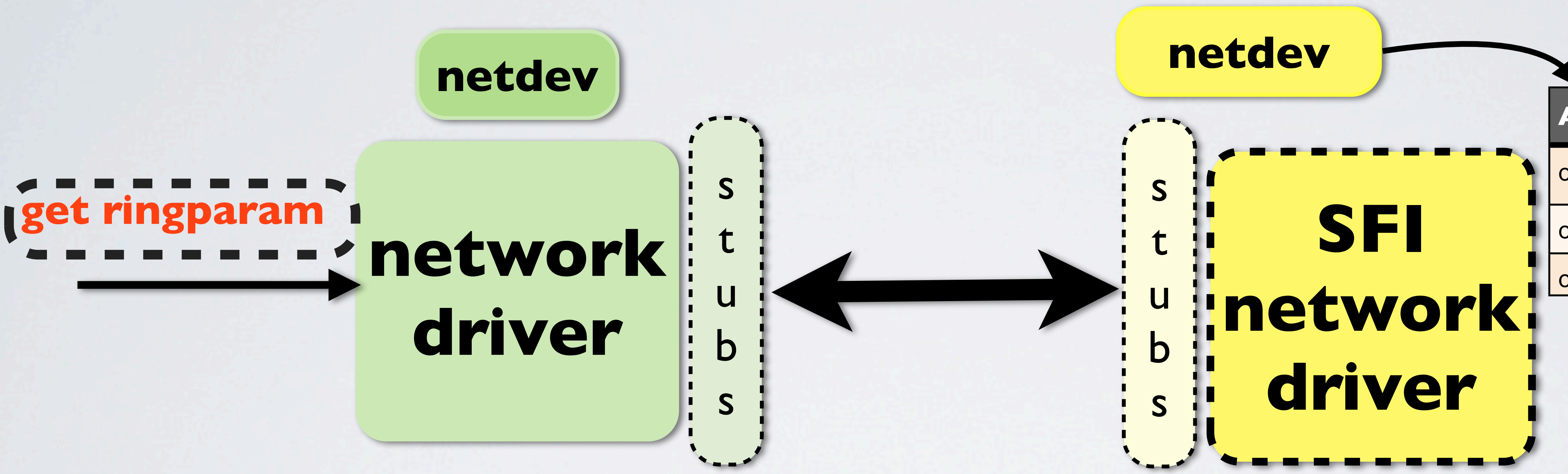
Transactional support through code generation



Transactional support through code generation



Transactional support through code generation

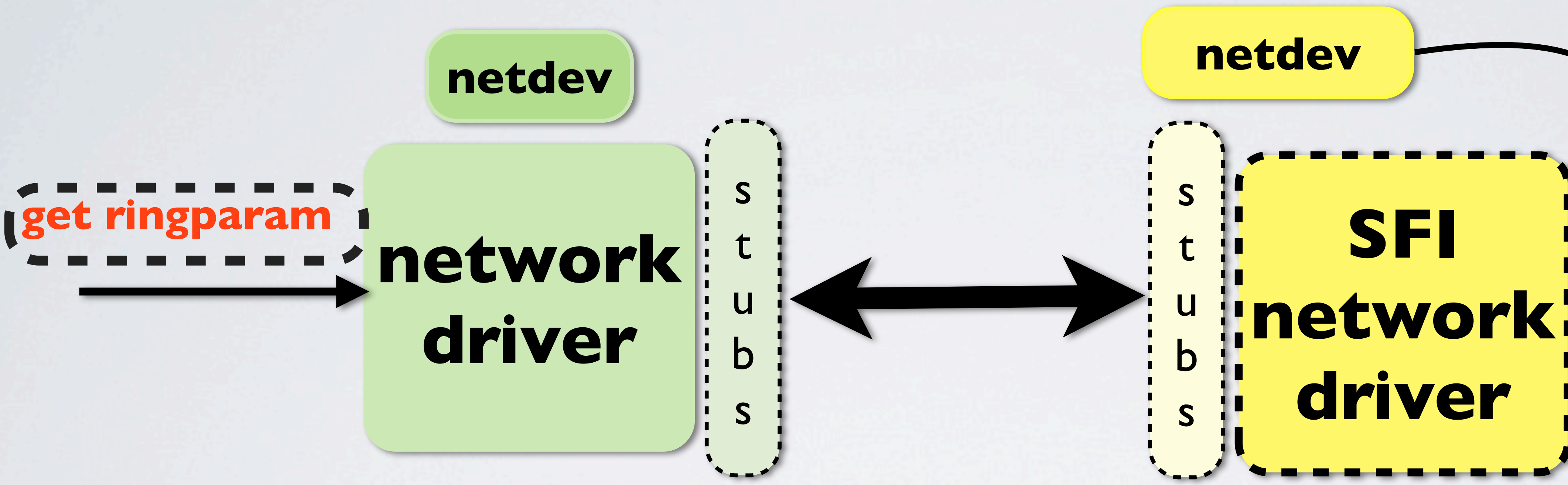


Range Table

Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

13

Transactional support through code generation

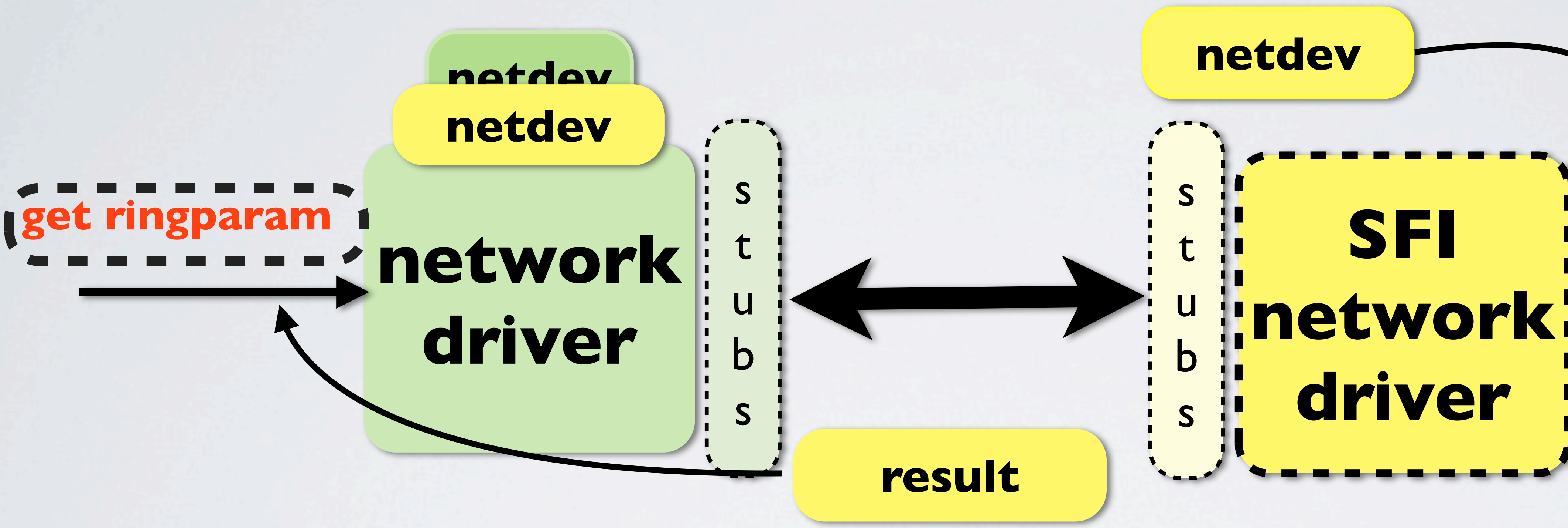


Range Table

Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

- **Detects and recovers from:**
  - **Memory errors like invalid 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

- ★ **Detects and recovers from:**
  - ★ **Memory errors like invalid pointer accesses**
  - ★ **Structural errors like malformed structures**
  - ★ **Processor exceptions like divide by zero, stack corruption**

13

Outline

Introduction

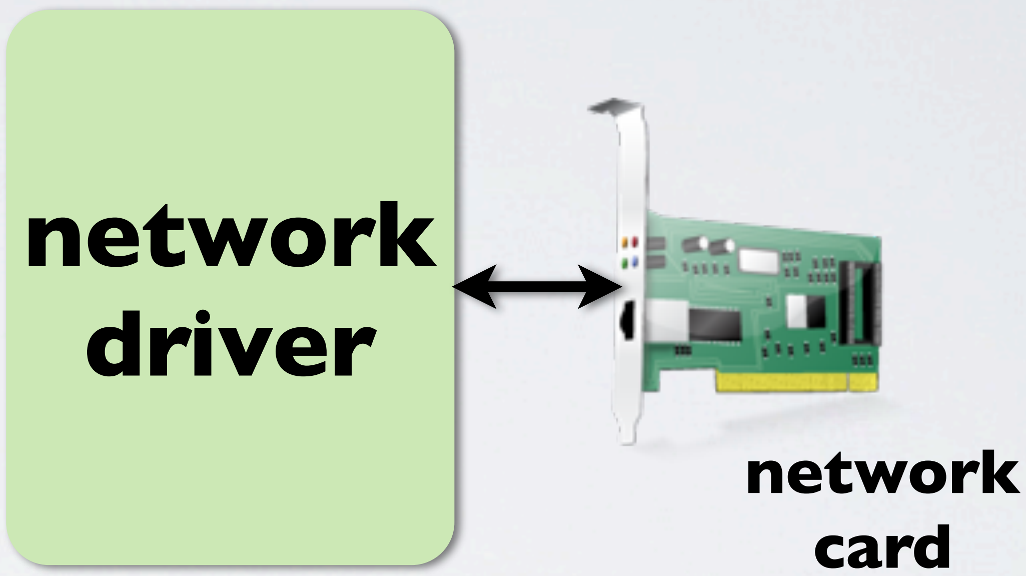
Fine-grained isolation

Checkpoint-based recovery

Conclusion

Checkpointing drivers is hard

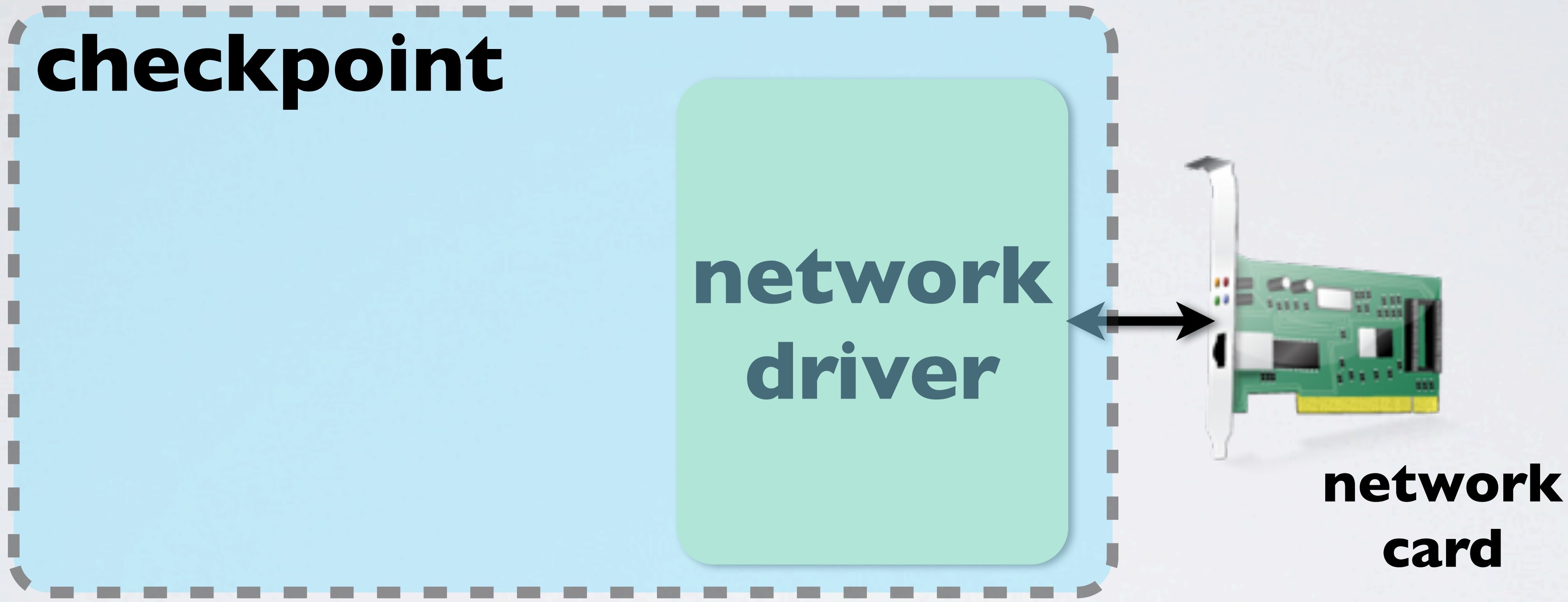
★ **Easy to capture memory state**



15

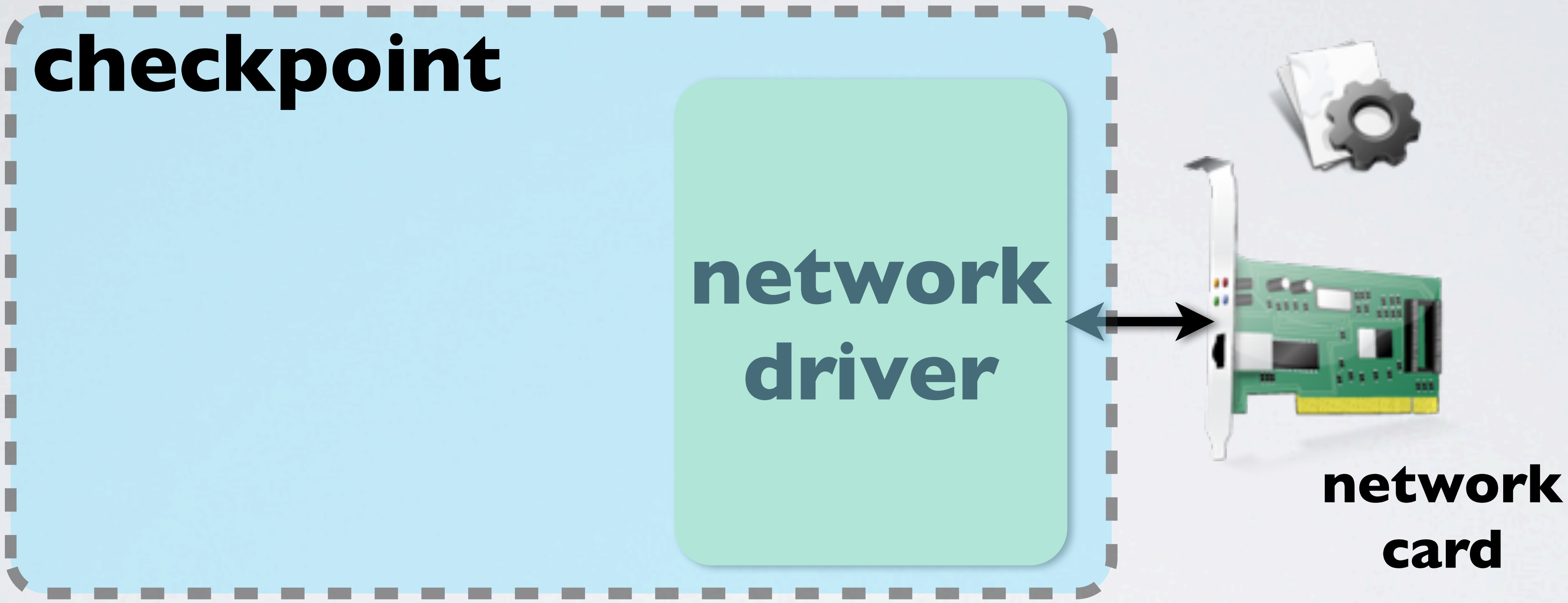
Checkpointing drivers is hard

★ **Easy to capture memory state**



Checkpointing drivers is hard

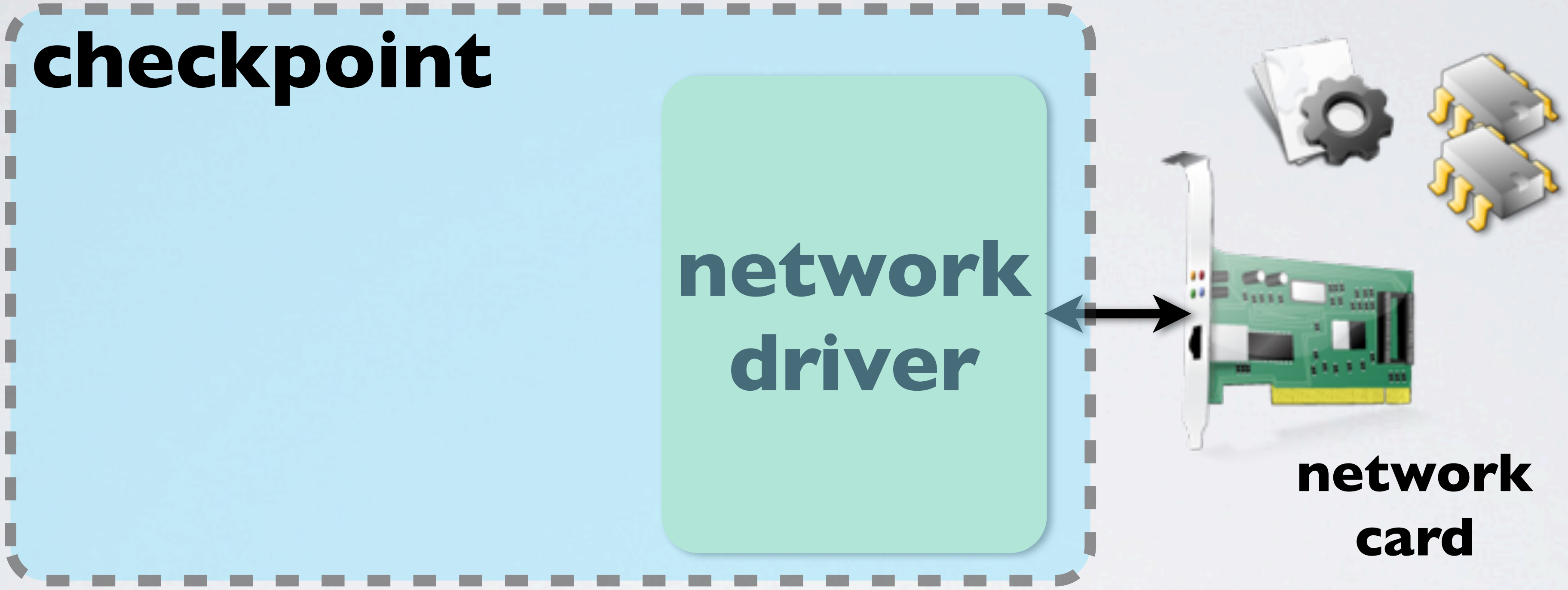
- ★ Easy to capture **memory** state



- ★ Device state is not captured
    - ★ **Device configuration space**

Checkpointing drivers is hard

★ **Easy to capture memory state**

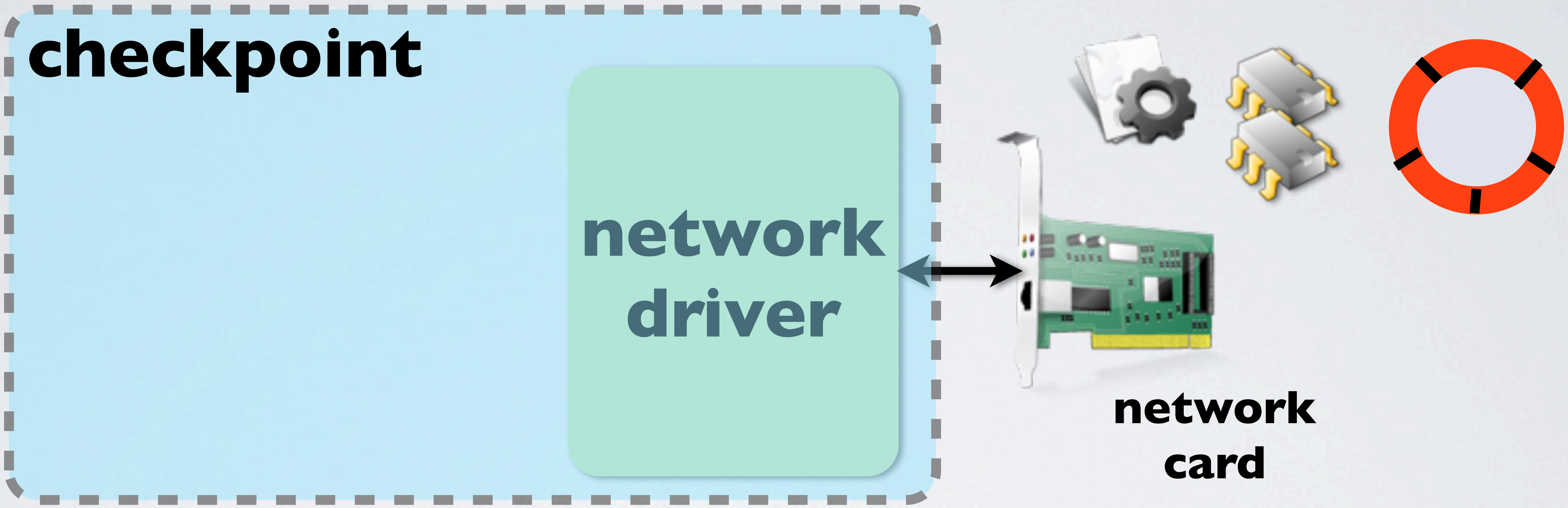


★ **Device state is not captured**

- ★ **Device configuration space**
- ★ **Internal device registers and counters**

Checkpointing drivers is hard

★ Easy to capture **memory** state

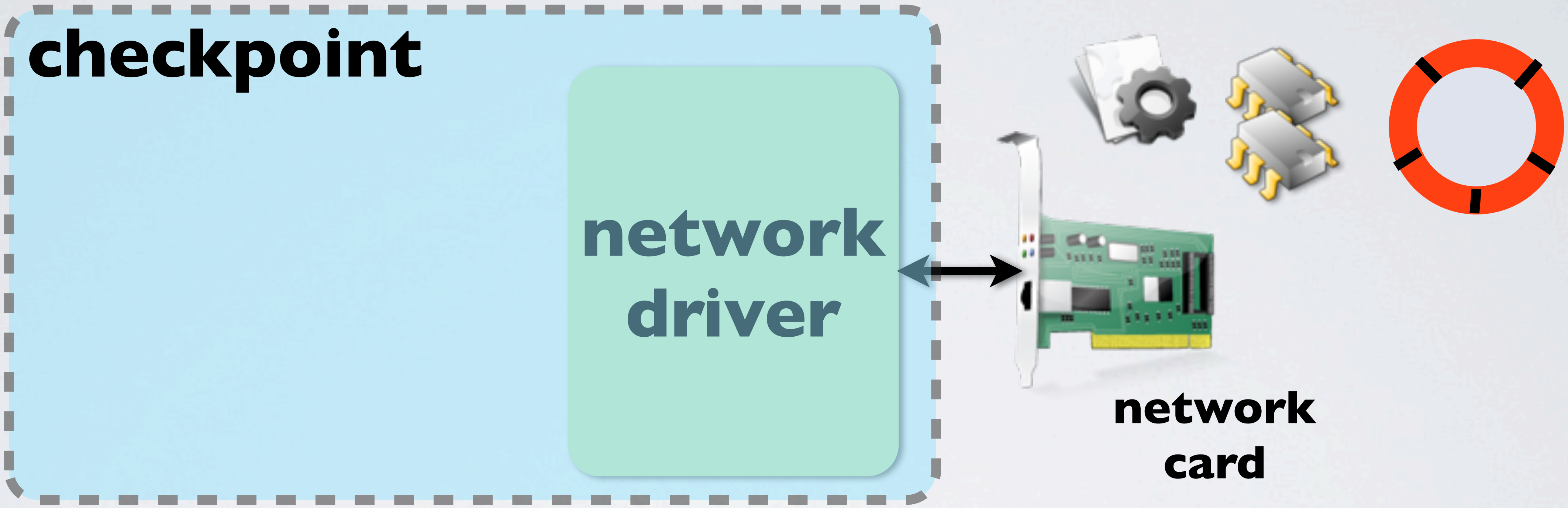


★ Device state is not captured

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

Checkpointing drivers is hard

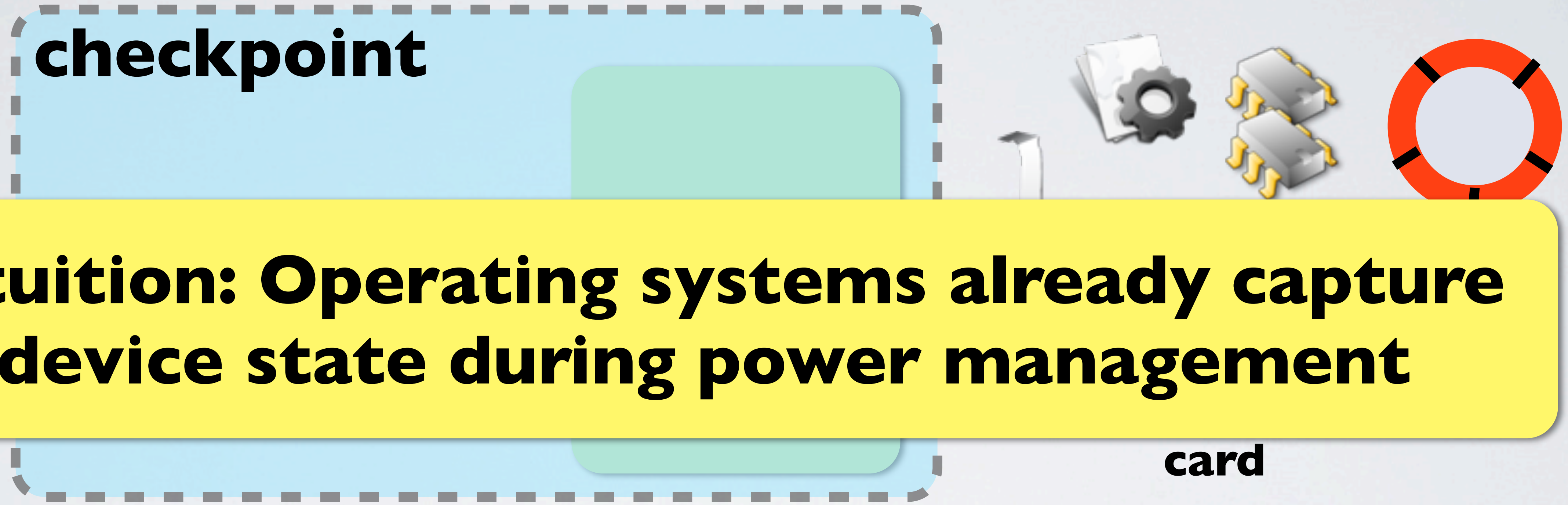
- Easy to capture **memory** state



- Device state is not captured
  - Device configuration space
  - Internal device registers and counters
  - Memory buffer addresses used for DMA
- Unique for every device

Checkpointing drivers is hard

- ★ **Easy to capture memory state**



Intuition: Operating systems already capture device state during power management

- ★ **Device state is not captured**
  - ★ **Device configuration space**
  - ★ **Internal device registers and counters**
  - ★ **Memory buffer addresses used for DMA**
- ★ **Unique for every device**

Intuition with power management

- Refactor power management code for device checkpoints
  - Correct: Developer captures unique device semantics
  - Fast: Avoids probe and latency critical for applications
- Ask developers to export checkpoint/restore in their drivers

Device checkpoint/restore from PM code

Suspend	Resume
Save config state	Restore config state
Save register state	Restore register state
Disable device	Restore or reset DMA state
Save DMA state	Re-attach/Enable device
Suspend device	Device Ready

Device checkpoint/restore from PM code

Suspend

- Save config state
- Save register state
- Save DMA state
- Suspend device

Resume

- Restore config state
- Restore register state
- Restore or reset DMA state
- Re-attach/Enable device
- Device Ready

17

Device checkpoint/restore from PM code

Suspend	Resume
Save config state	Restore config state
Save register state	Restore register state
	Restore or reset DMA state
Save DMA state	Re-attach/Enable device
	Device Ready

Device checkpoint/restore from PM code

Suspend

- Save config state
- Save register state
- Save DMA state

Resume

- Restore config state
- Restore register state
- Restore or reset DMA state
- Re-attach/Enable device
- Device Ready

17

Device checkpoint/restore from PM code

Checkpoint

Resume

Checkpoint	Resume
Save config state	Restore config state
Save register state	Restore register state
Save DMA state	Restore or reset DMA state
	Re-attach/Enable device
	Device Ready

17

Device checkpoint/restore from PM code

Checkpoint	Resume
Save config state	Restore config state
Save register state	Restore register state
Save DMA state	Restore or reset DMA state
	Re-attach/Enable device

Device checkpoint/restore from PM code

Checkpoint

Save config state

Save register state

Save DMA state

Resume

Restore config state

Restore register state

Restore or reset DMA state

Device checkpoint/restore from PM code

Checkpoint

Save config state
Save register state
Save DMA state

Restore

Restore config state
Restore register state
Restore or reset DMA state

Device checkpoint/restore from PM code

Checkpoint

Save config state
Save register state
Save DMA state

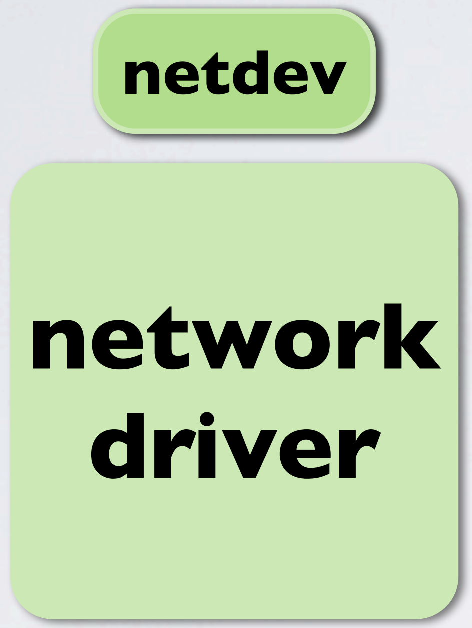
Restore

Restore config state
Restore register state
Restore or reset DMA state

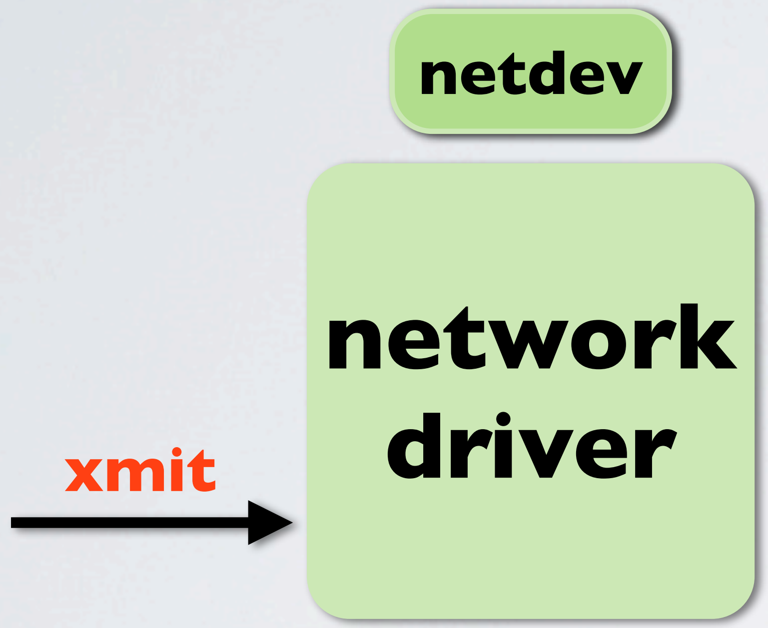
Suspend/resume code provides device checkpoint functionality

17

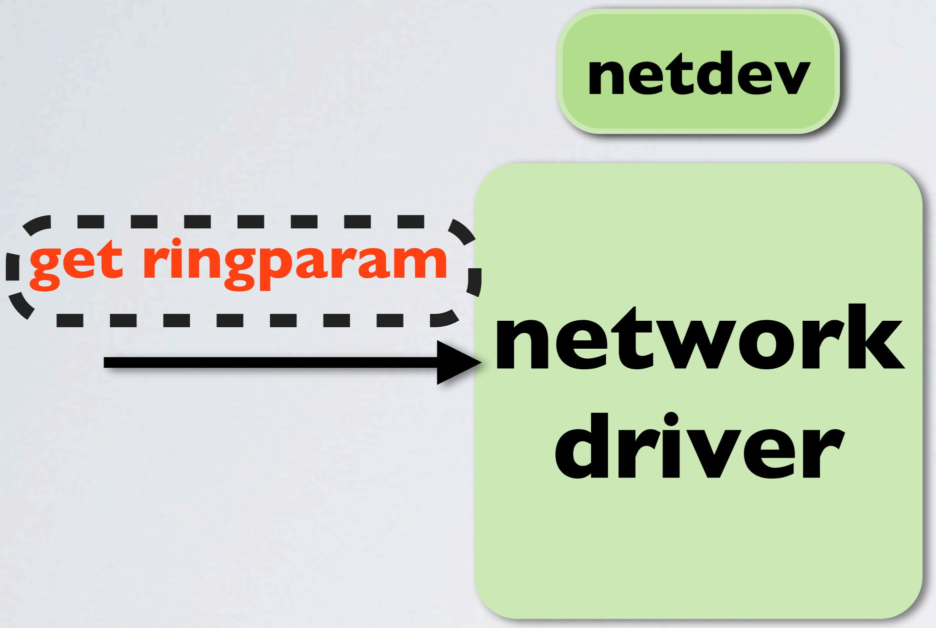
Synergy of isolation and fast checkpoints



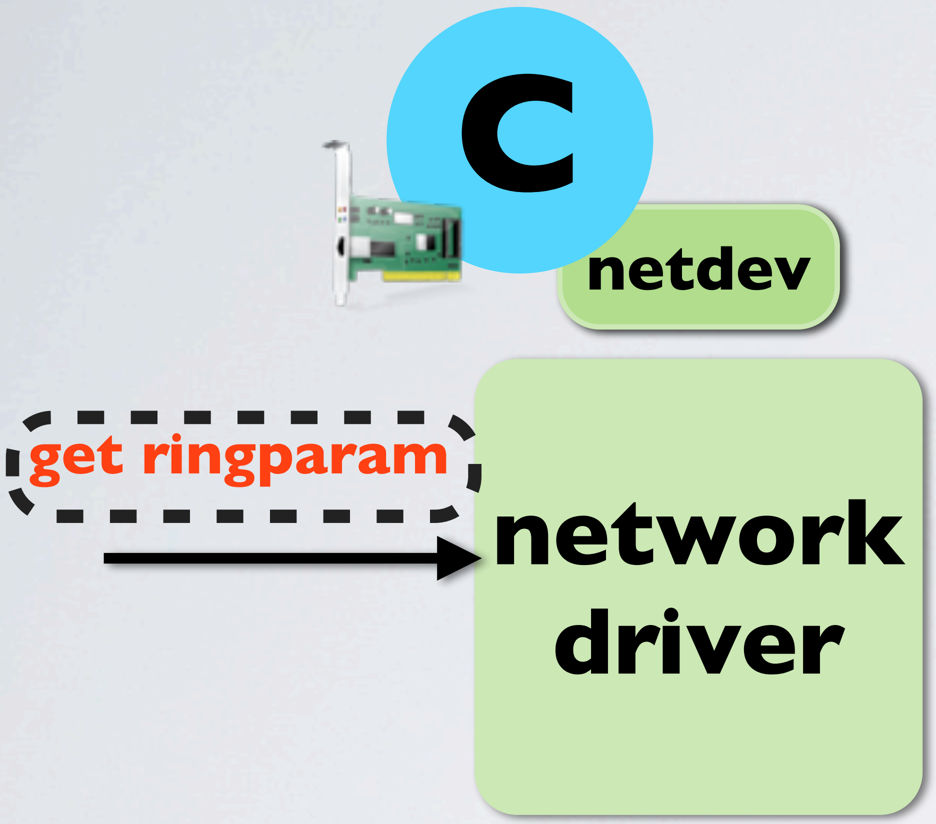
Synergy of isolation and fast checkpoints



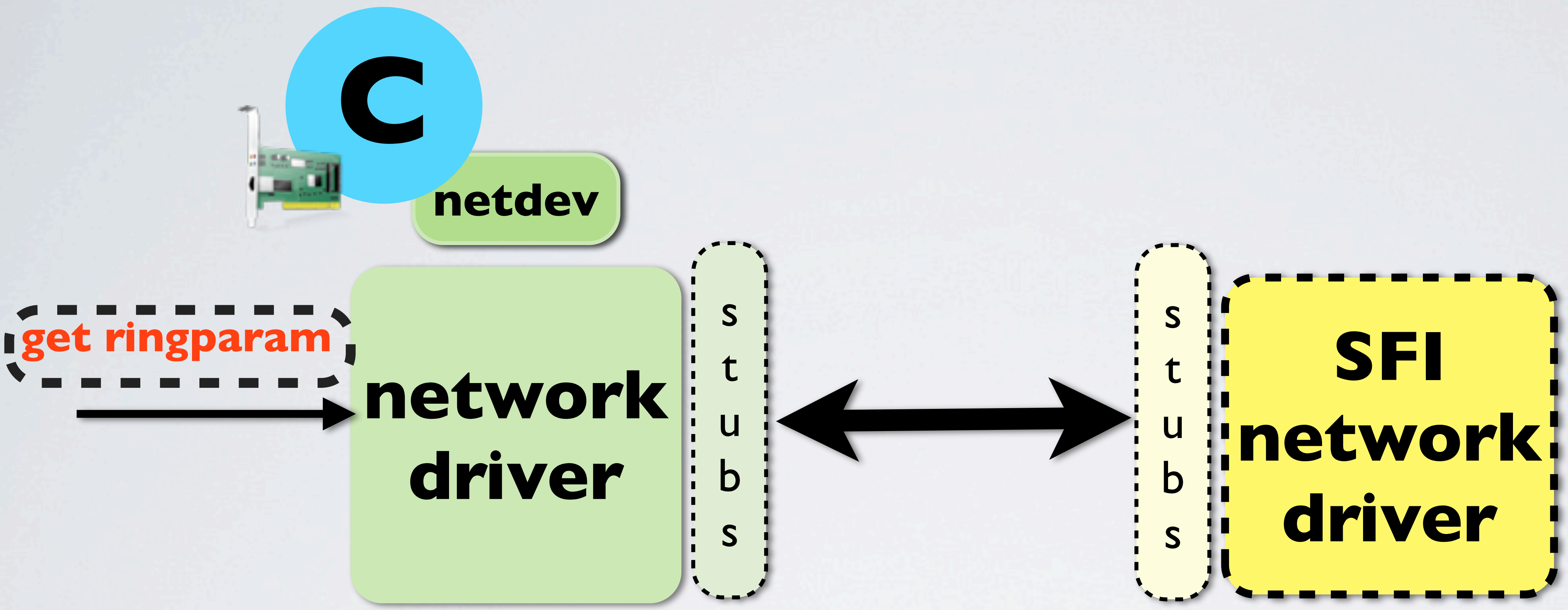
Synergy of isolation and fast checkpoints



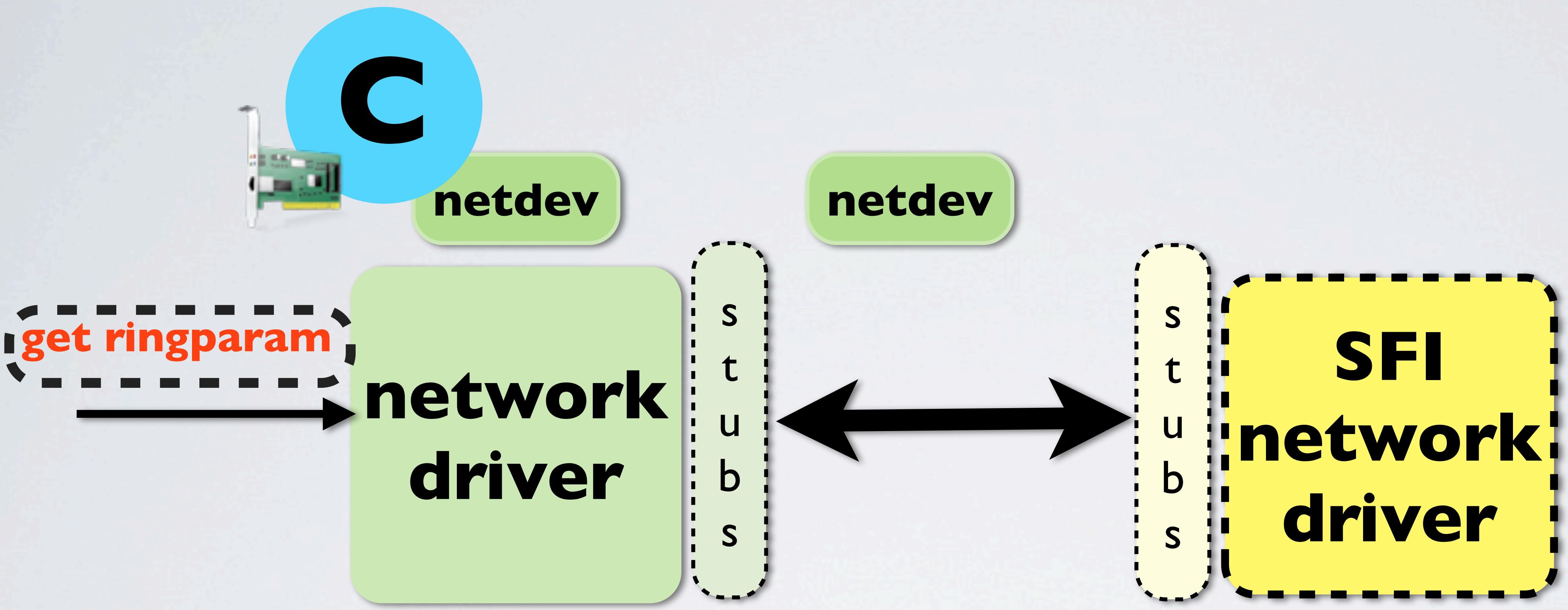
Synergy of isolation and fast checkpoints



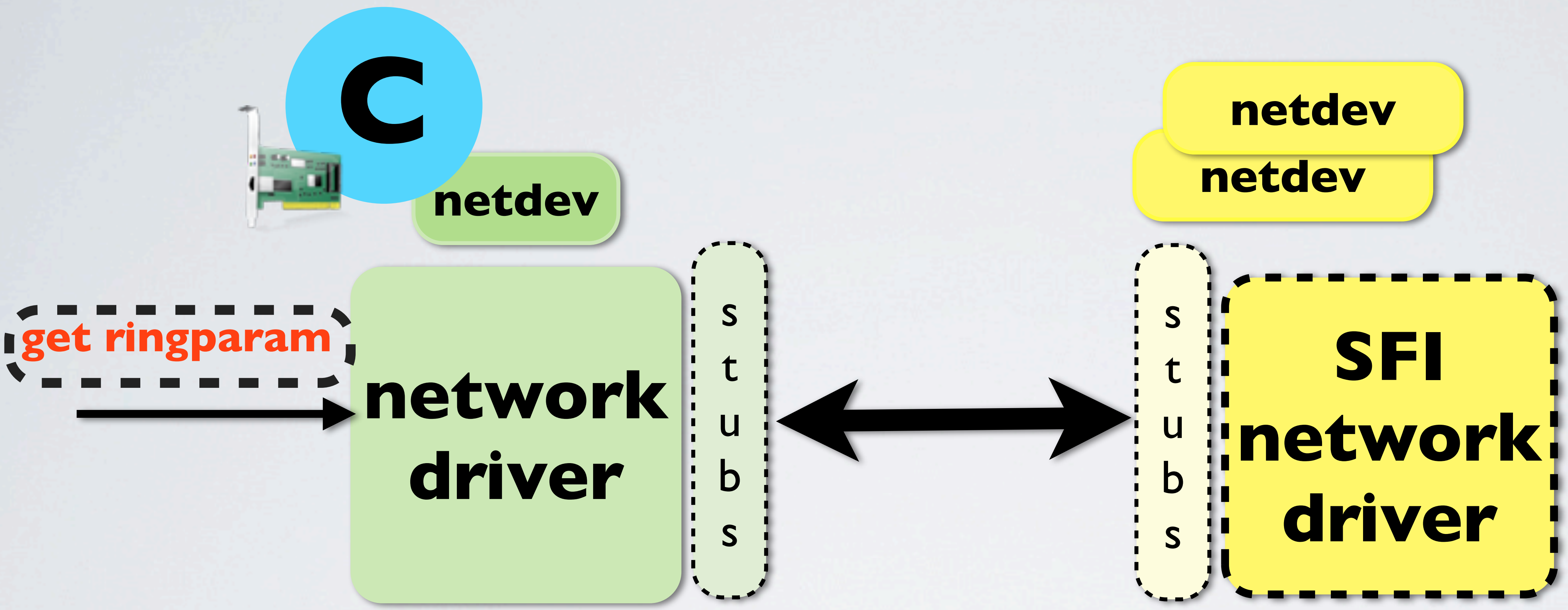
Synergy of isolation and fast checkpoints



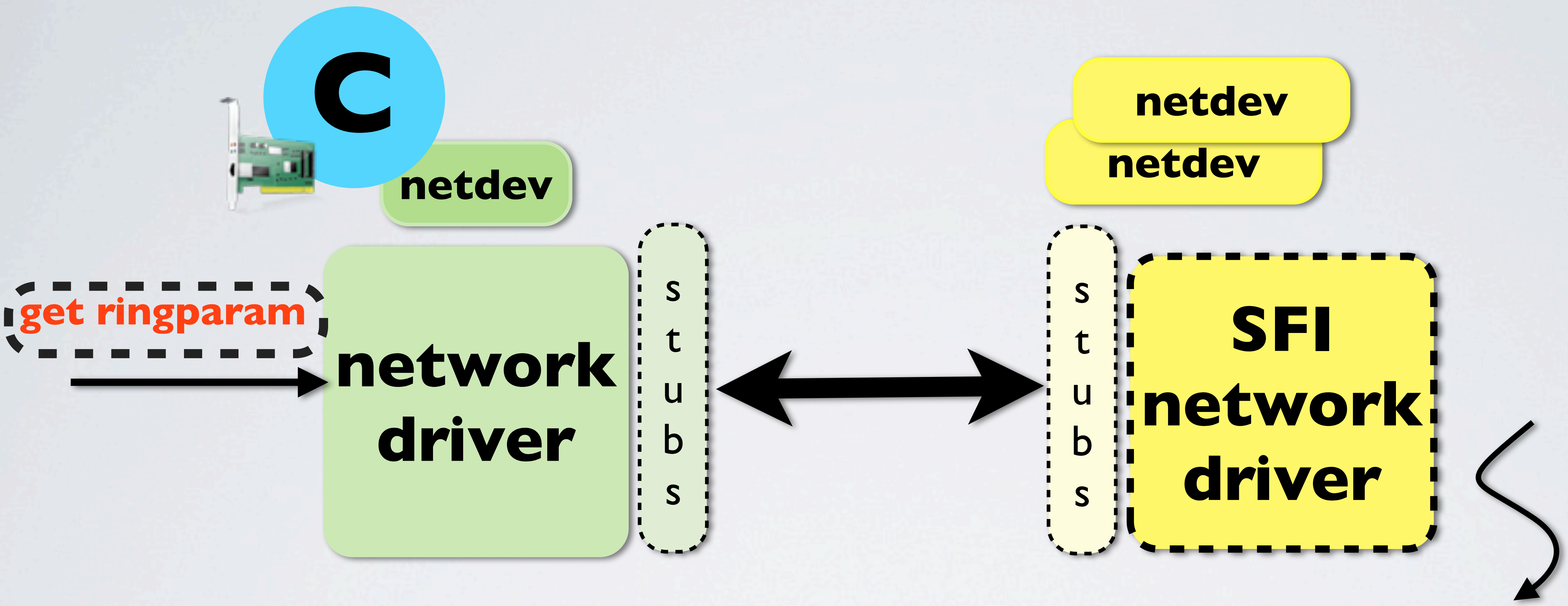
Synergy of isolation and fast checkpoints



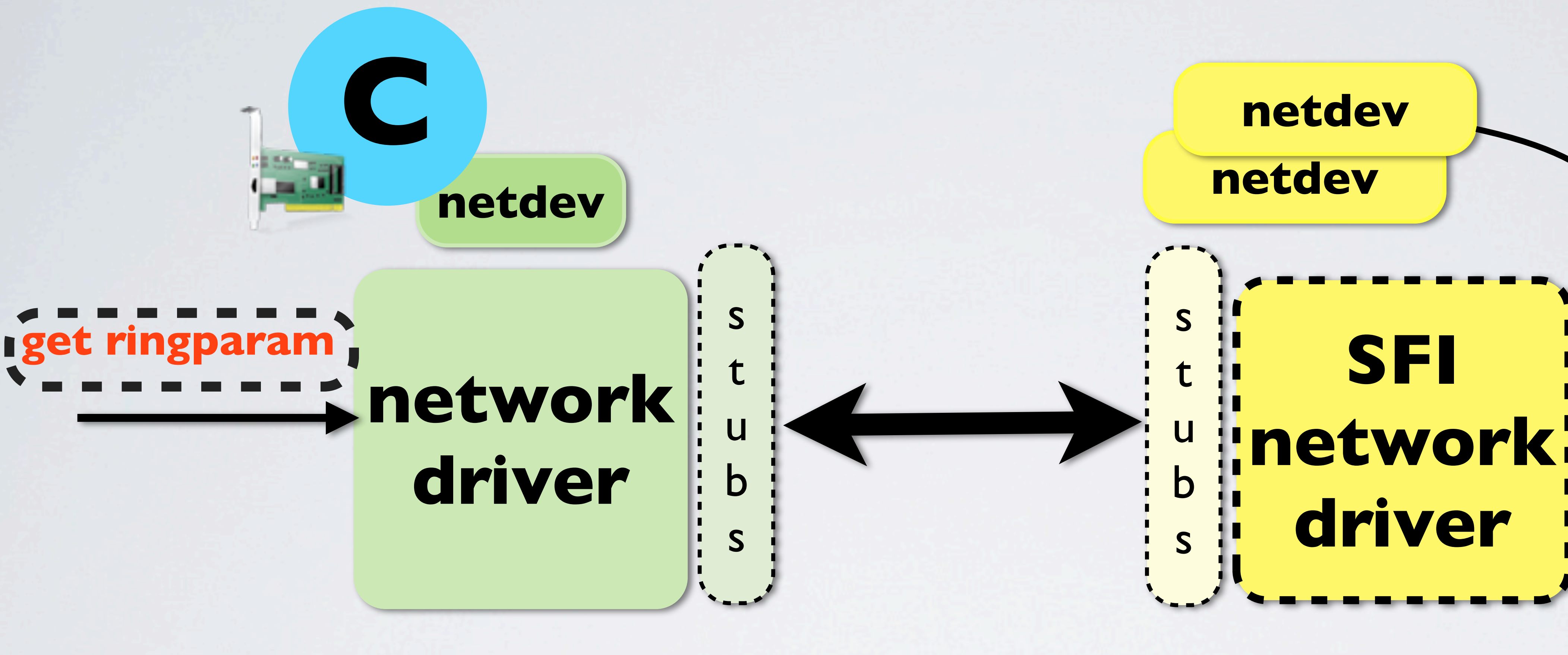
Synergy of isolation and fast checkpoints



Synergy of isolation and fast checkpoints

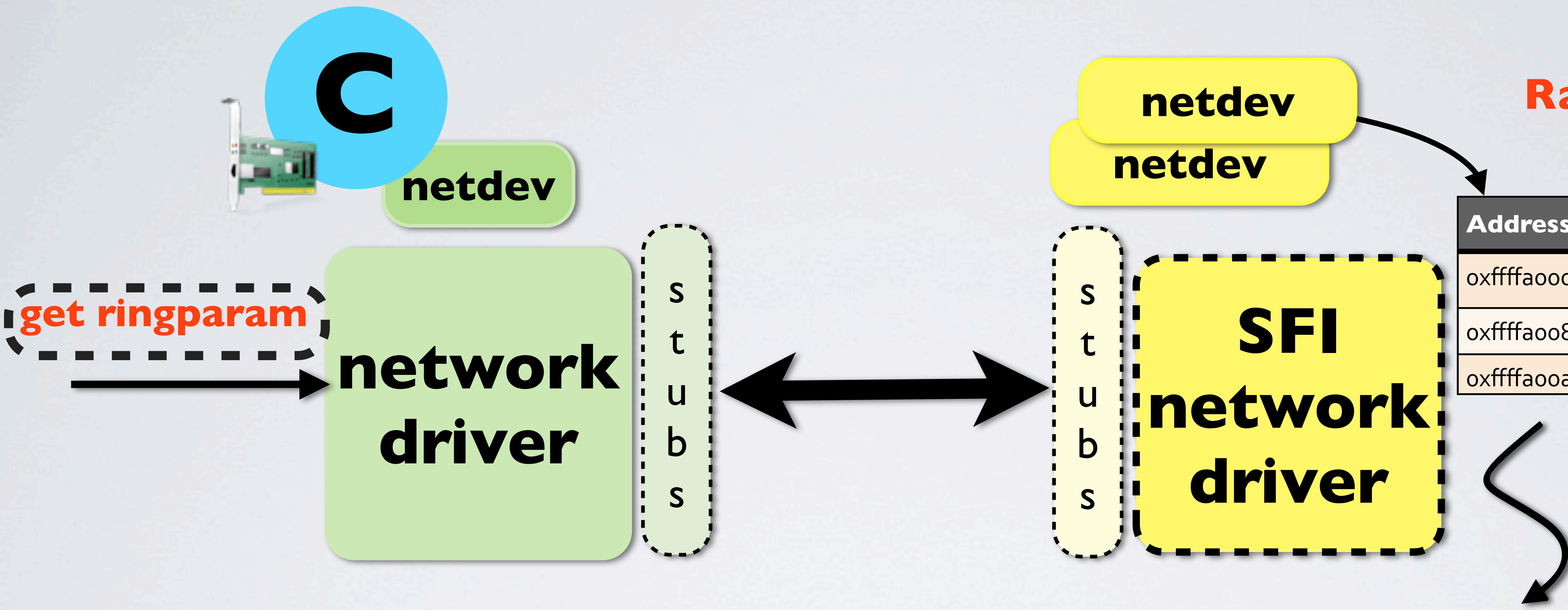


Synergy of isolation and fast checkpoints



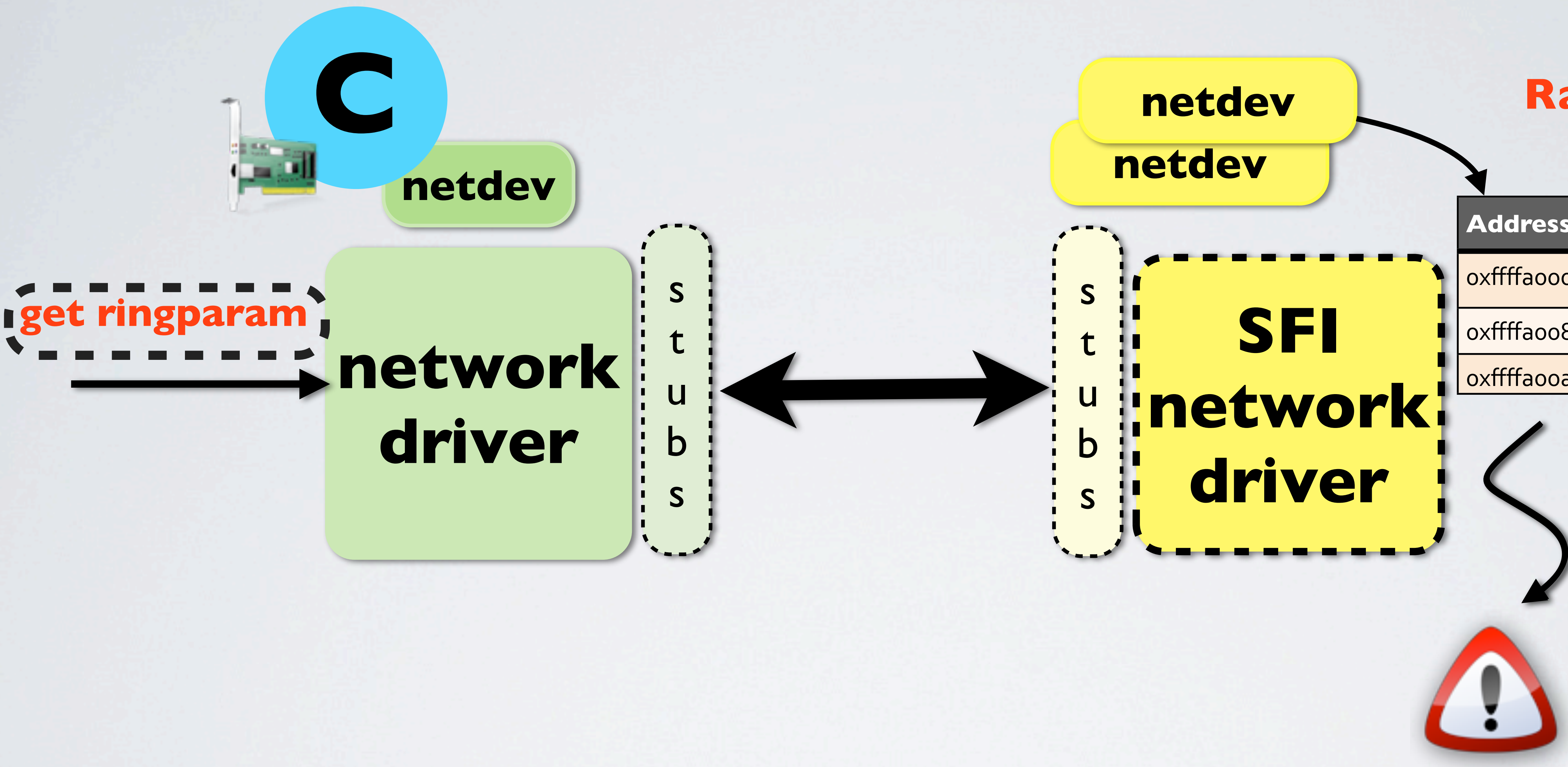
Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

Synergy of isolation and fast checkpoints



Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

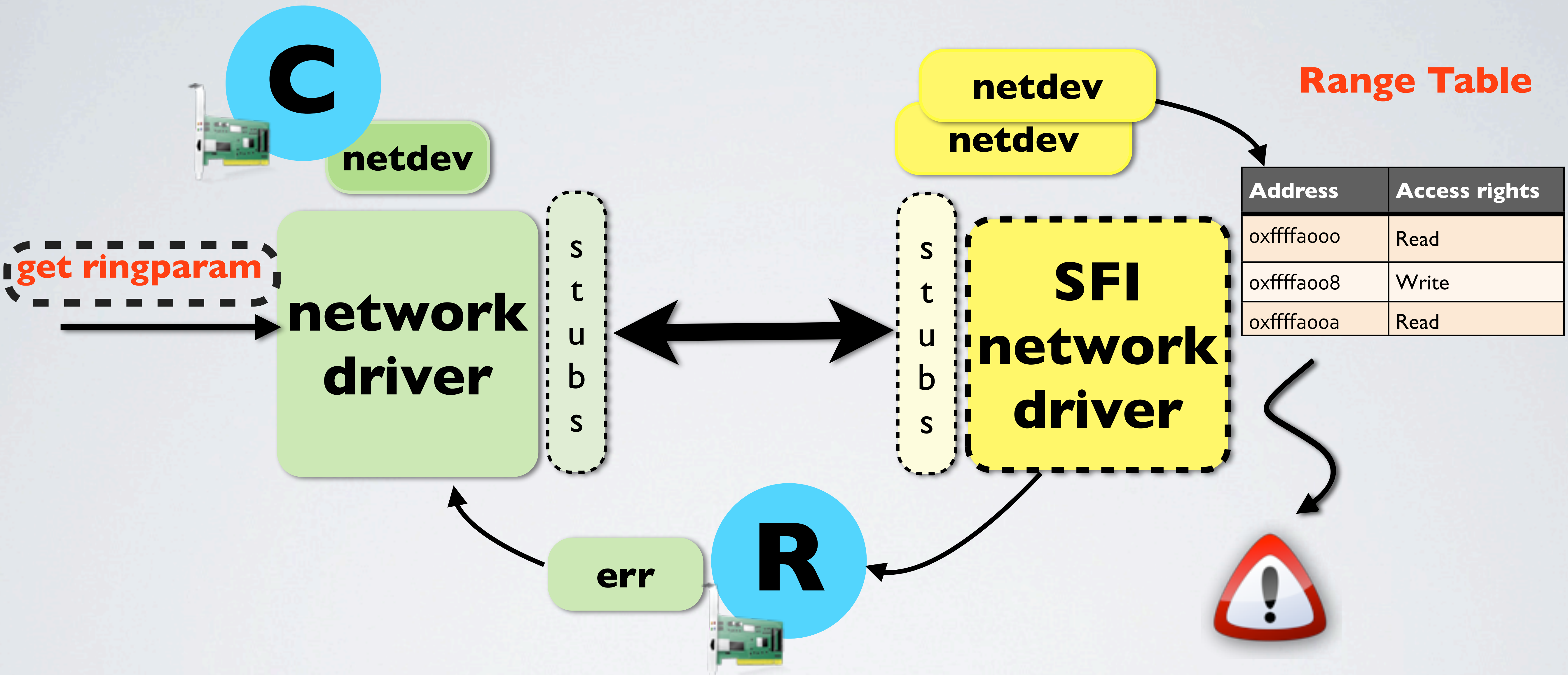
Synergy of isolation and fast checkpoints



Range Table

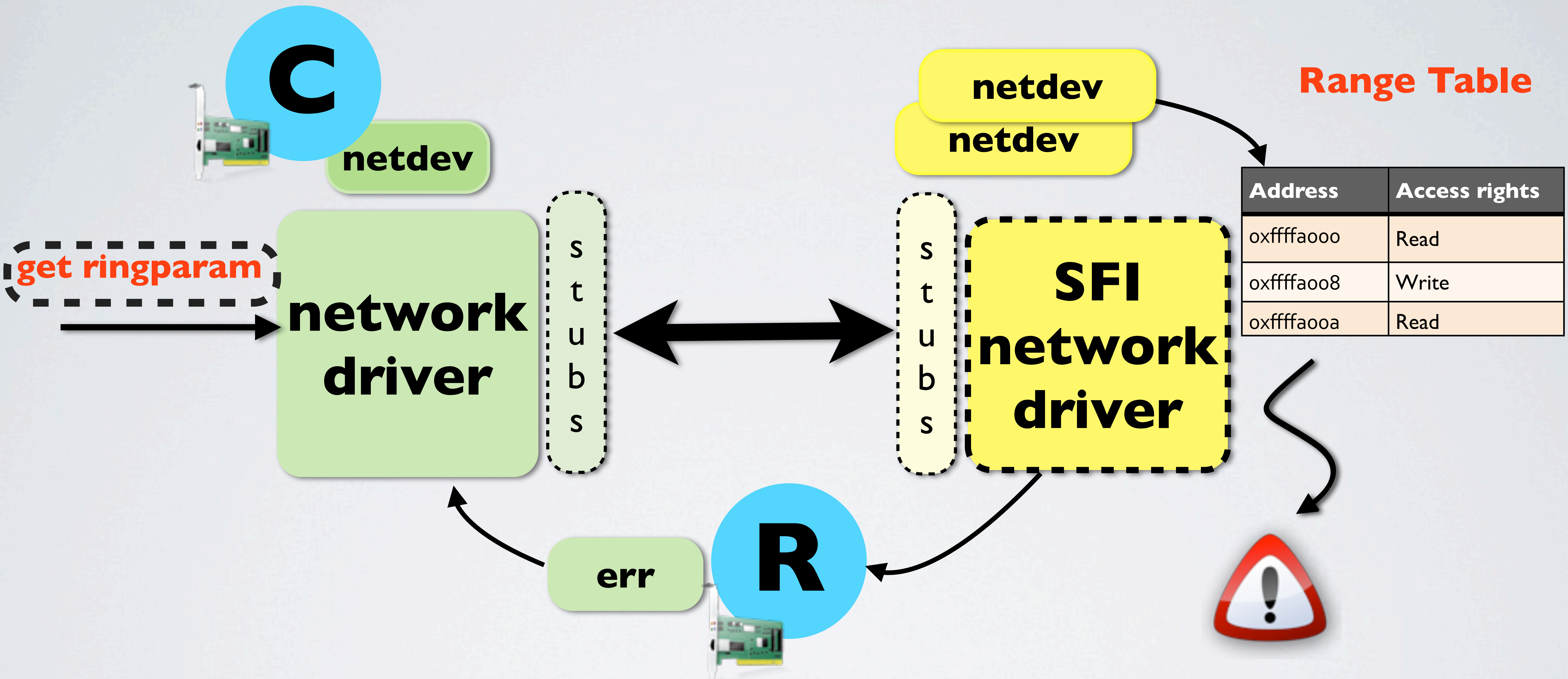
Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

Synergy of isolation and fast checkpoints



Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

Synergy of isolation and fast checkpoints



Address	Access rights
oxffffaooo	Read
oxffffaoo8	Write
oxffffaooa	Read

FGFT provides transactional execution of driver entry points

How does this give us transactional execution?

How does this give us transactional execution?

- **Atomicity: All or nothing execution**
  - **Driver state: Run code in SFI module**
  - **Device state: Explicitly checkpoint/restore state**

How does this give us transactional execution?

- **Atomicity: All or nothing execution**
  - **Driver state: Run code in SFI module**
  - **Device state: Explicitly checkpoint/restore state**

- **Isolation: Serialization to hide incomplete transactions**
  - **Re-use existing device locks to lock driver**
  - **Two phase locking**

How does this give us transactional execution?

- **Atomicity: All or nothing execution**
  - **Driver state: Run code in SFI module**
  - **Device state: Explicitly checkpoint/restore state**
- **Isolation: Serialization to hide incomplete transactions**
  - **Re-use existing device locks to lock driver**
  - **Two phase locking**
- **Consistency: Only valid (kernel, driver and device) states**
  - **Higher level mechanisms to rollback external actions**
  - **At most once device action guarantee to applications**

Outline

Introduction

Fine-grained isolation

Checkpoint-based recovery

Evaluation & Conclusions

Evaluation platform

- **Criterion :**
  - **Latency of recovery: How fast is it?**
  - **Correctness of recovery: How well does it work?**
  - **Incremental effort: How much work is it?**
  - **Performance: How much does it cost?**

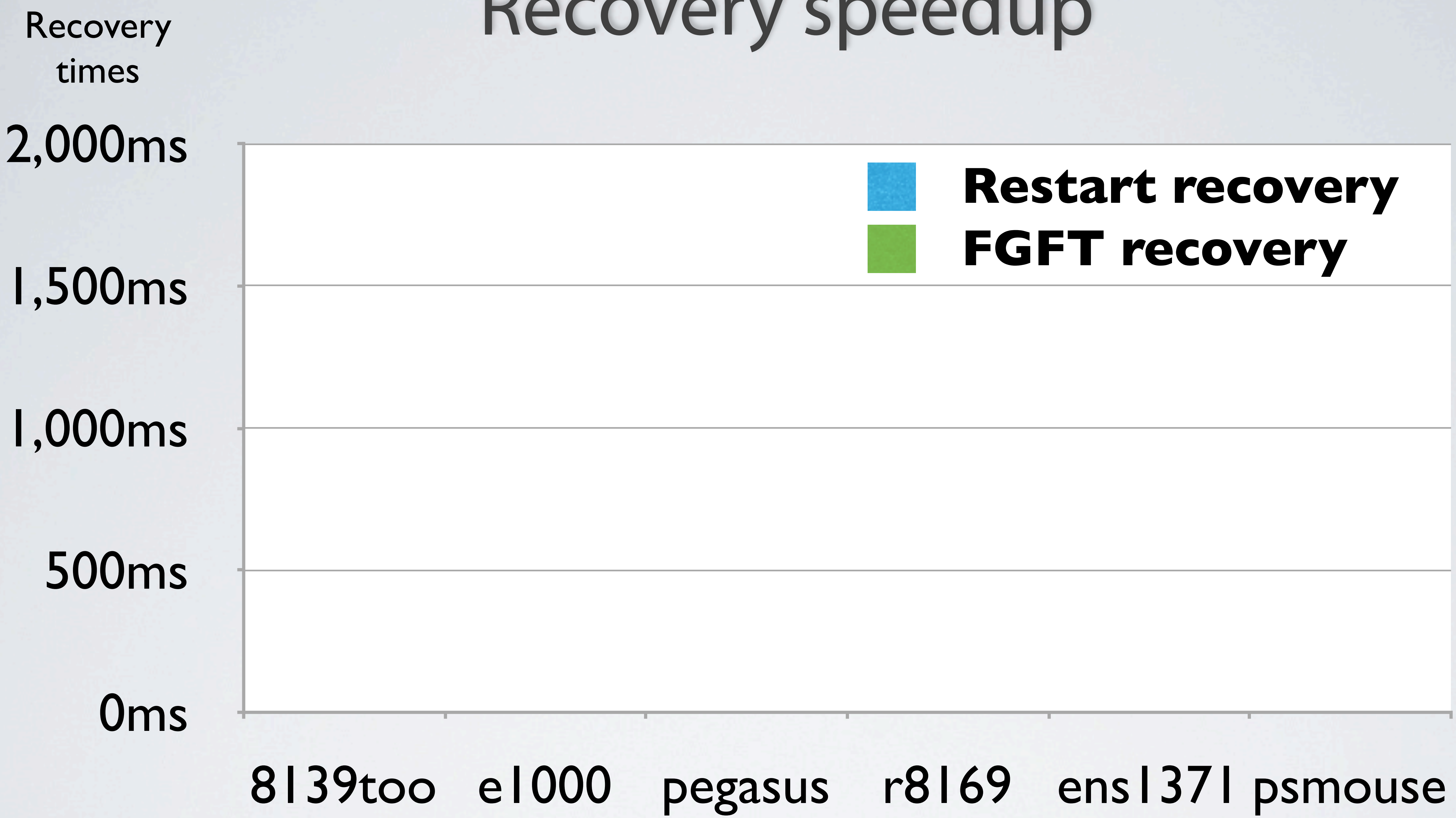
Evaluation platform

- Criterion :
  - Latency of recovery: How fast is it?
  - Correctness of recovery: How well does it work?
  - Incremental effort: How much work is it?
  - Performance: How much does it cost?

- Platform :
  - Implemented in Linux 2.6.29
  - 2.5 GHz Intel Core 2 Quad core w/ 4 GB DDR2 DRAM
  - Six drivers across three classes

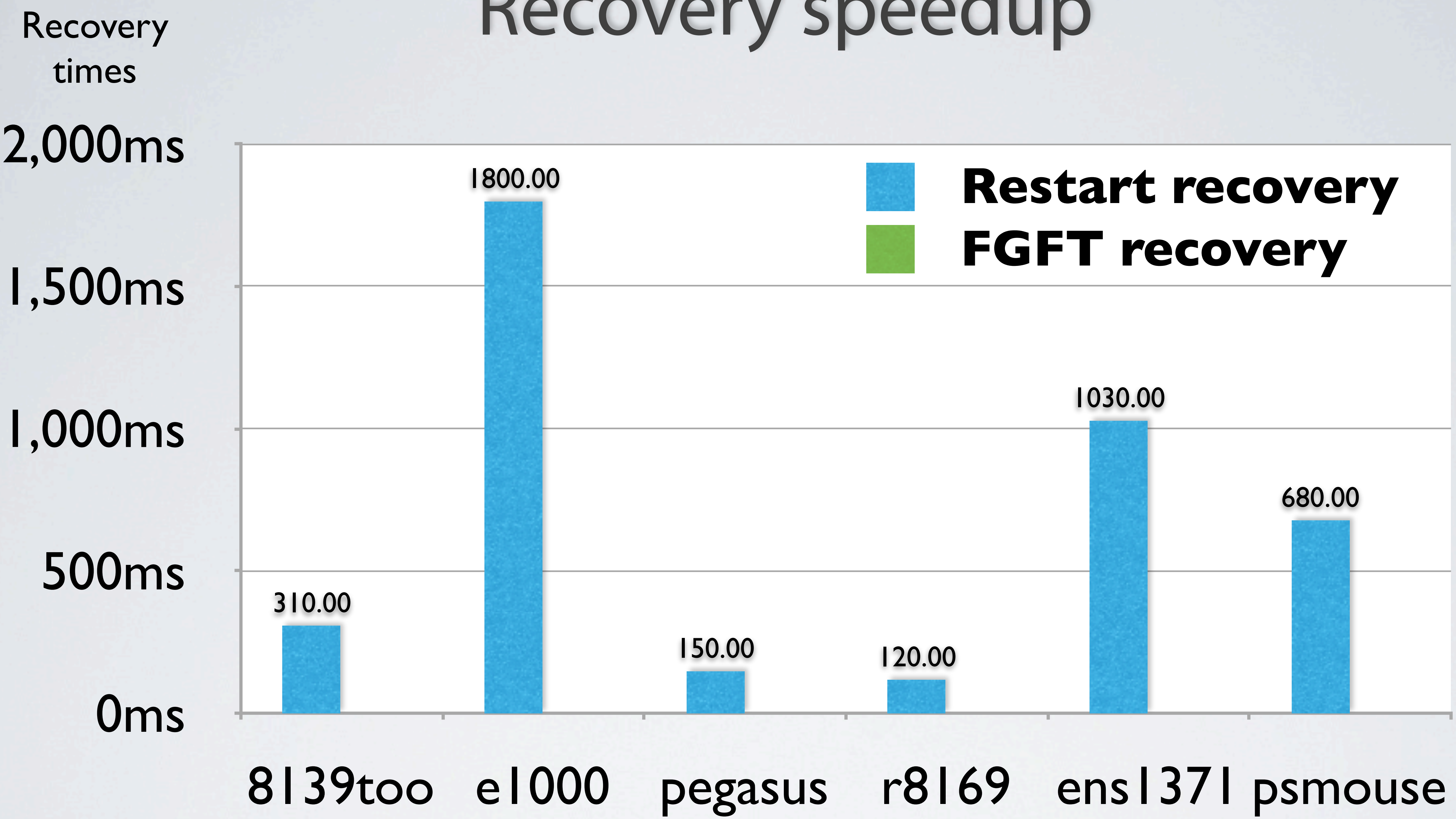
Driver	Class	Bus
8139too	net	PCI
e1000	net	PCI
r8169	net	PCI
pegasus	net	USB
psmouse	sound	PCI
ens1371	input	serio

Recovery speedup



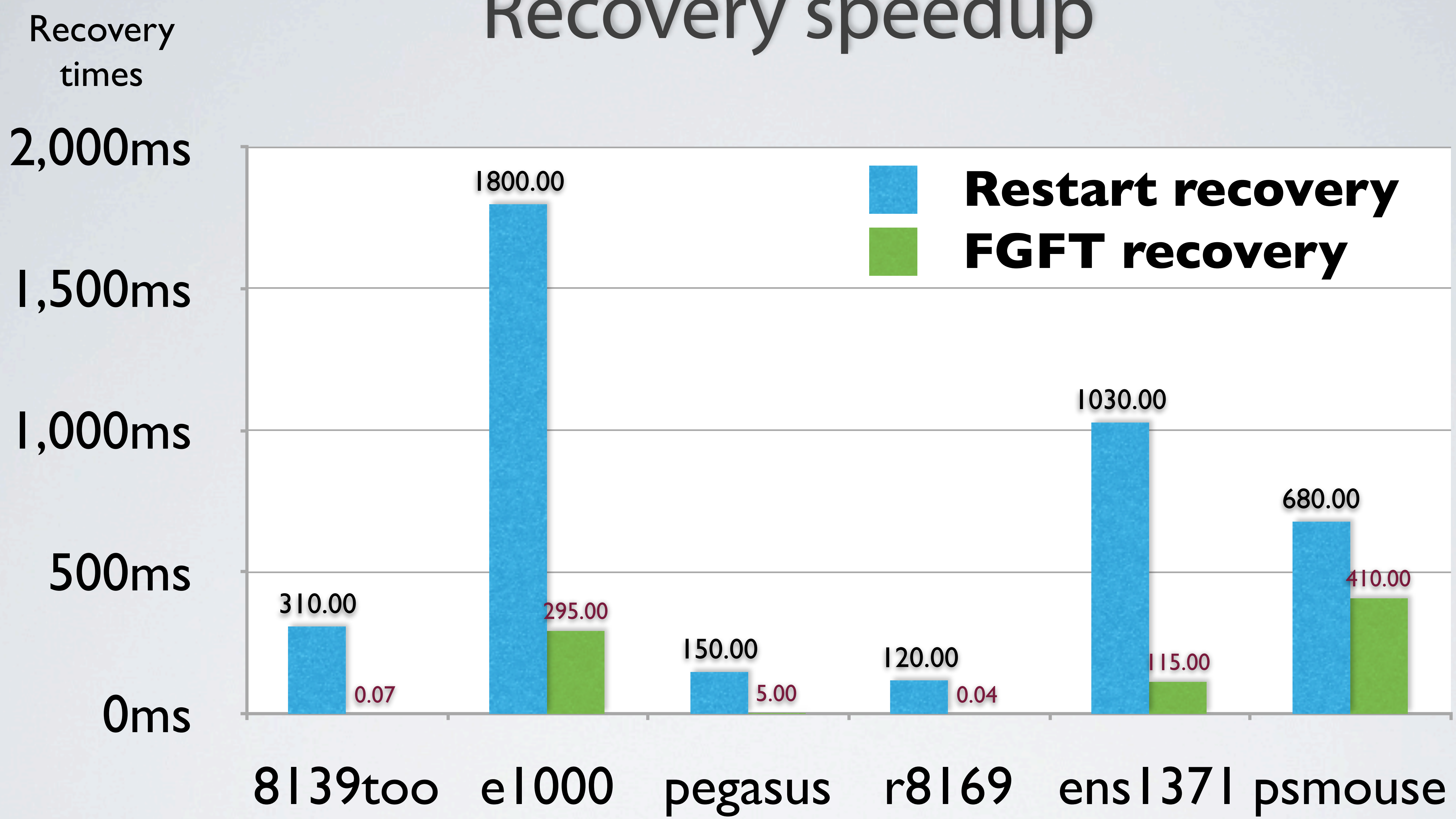
22

Recovery speedup

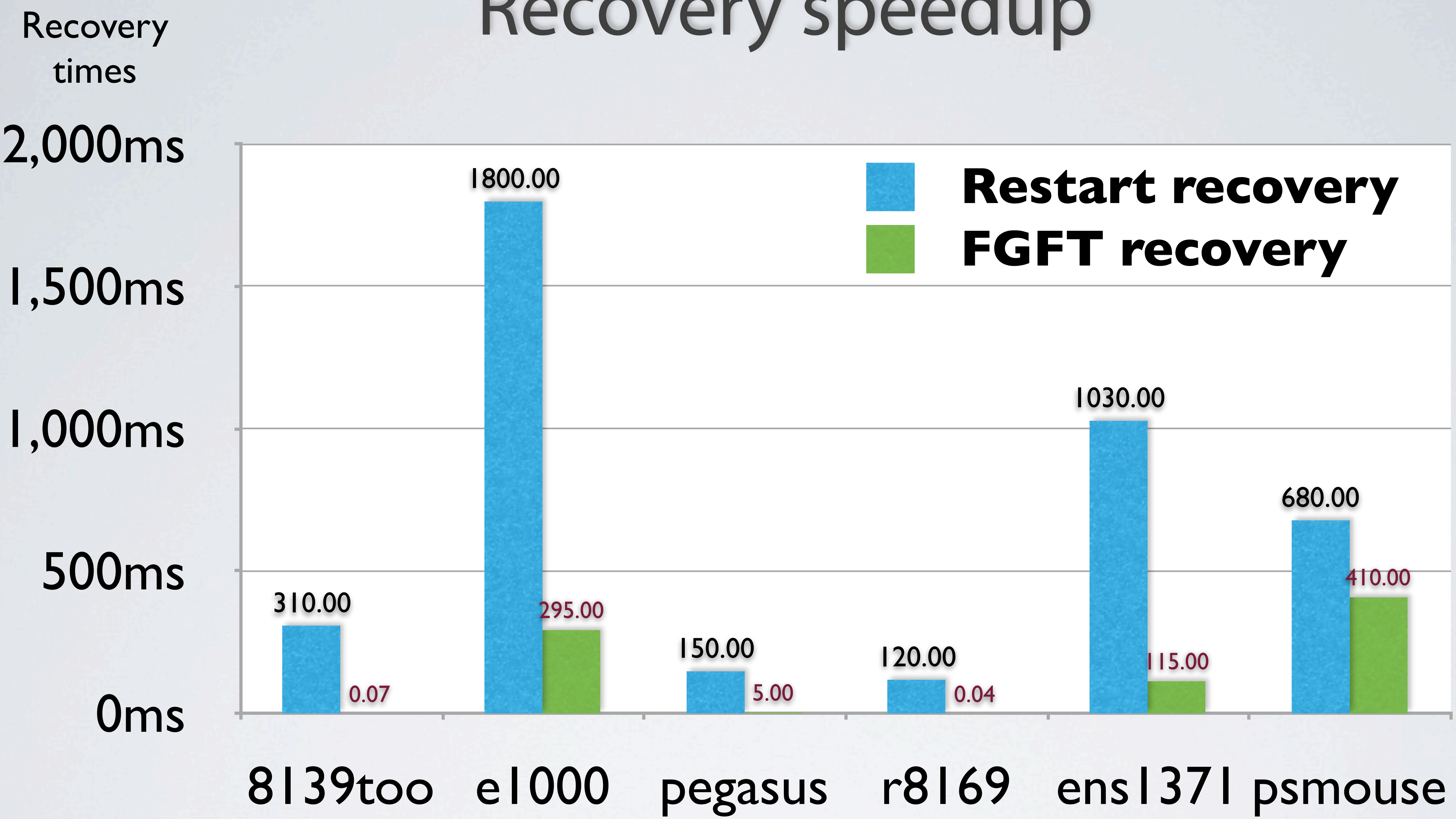


22

Recovery speedup



Recovery speedup



FGFT provides significant speedup in driver recovery and improves system availability

Static and dynamic fault injection

Driver	Injected Faults	Native Crashes
8139too	43	43
e1000	47	47
r8169	36	36
pegasus	34	33
ens1371	22	21
psmouse	46	46
TOTAL	258	256

Static and dynamic fault injection

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

Static and dynamic fault injection

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

FGFT recovers from multiple failures : 1) restores non-class state and 2) does not affect other threads

23

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

24

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

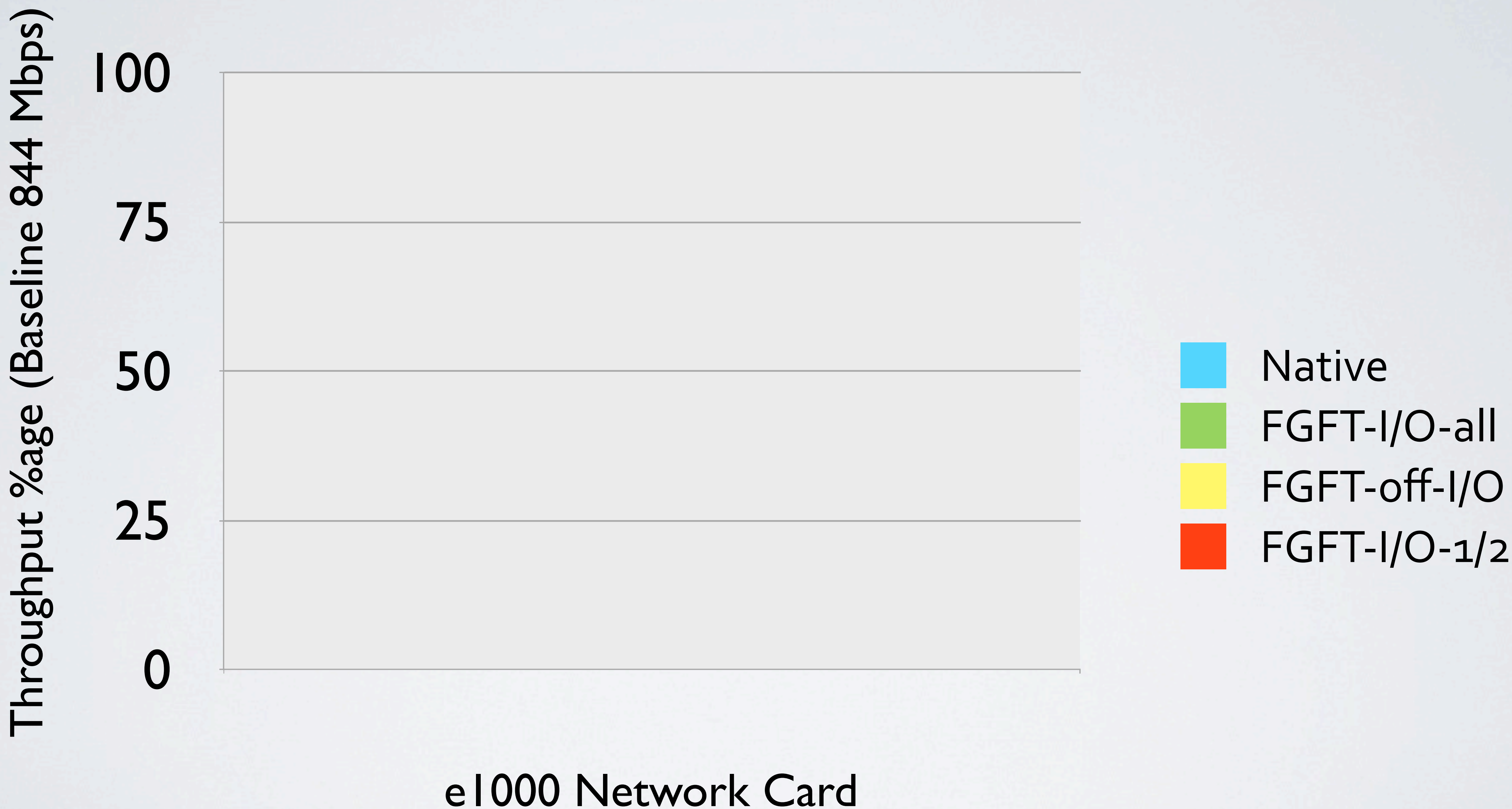
FGFT requires a loadable kernel module (1200 LOC) and 38 lines of kernel changes to trap processor exceptions

Throughput with isolation and recovery

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

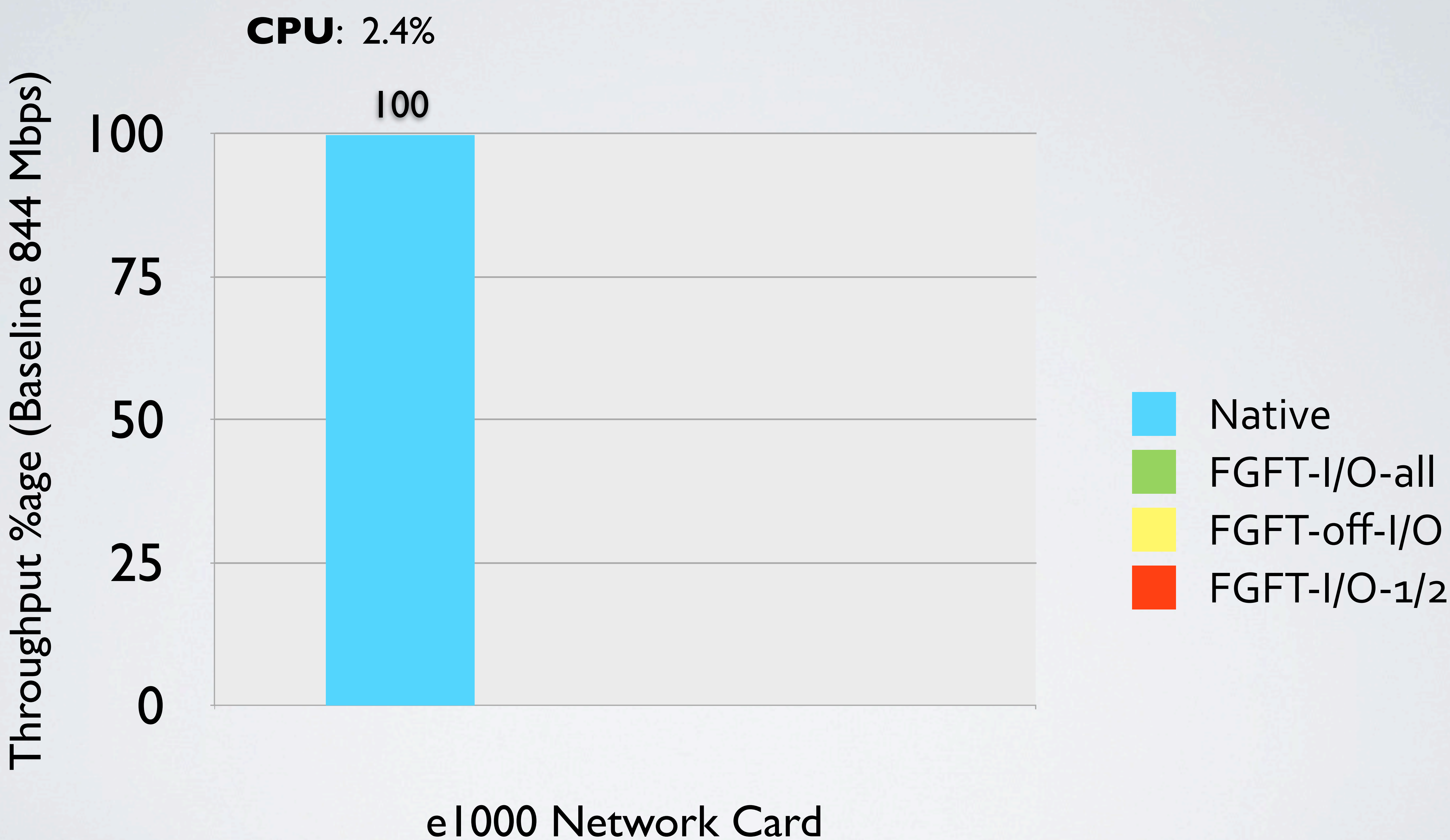
netperf on Intel quad-core machines

Throughput with isolation and recovery



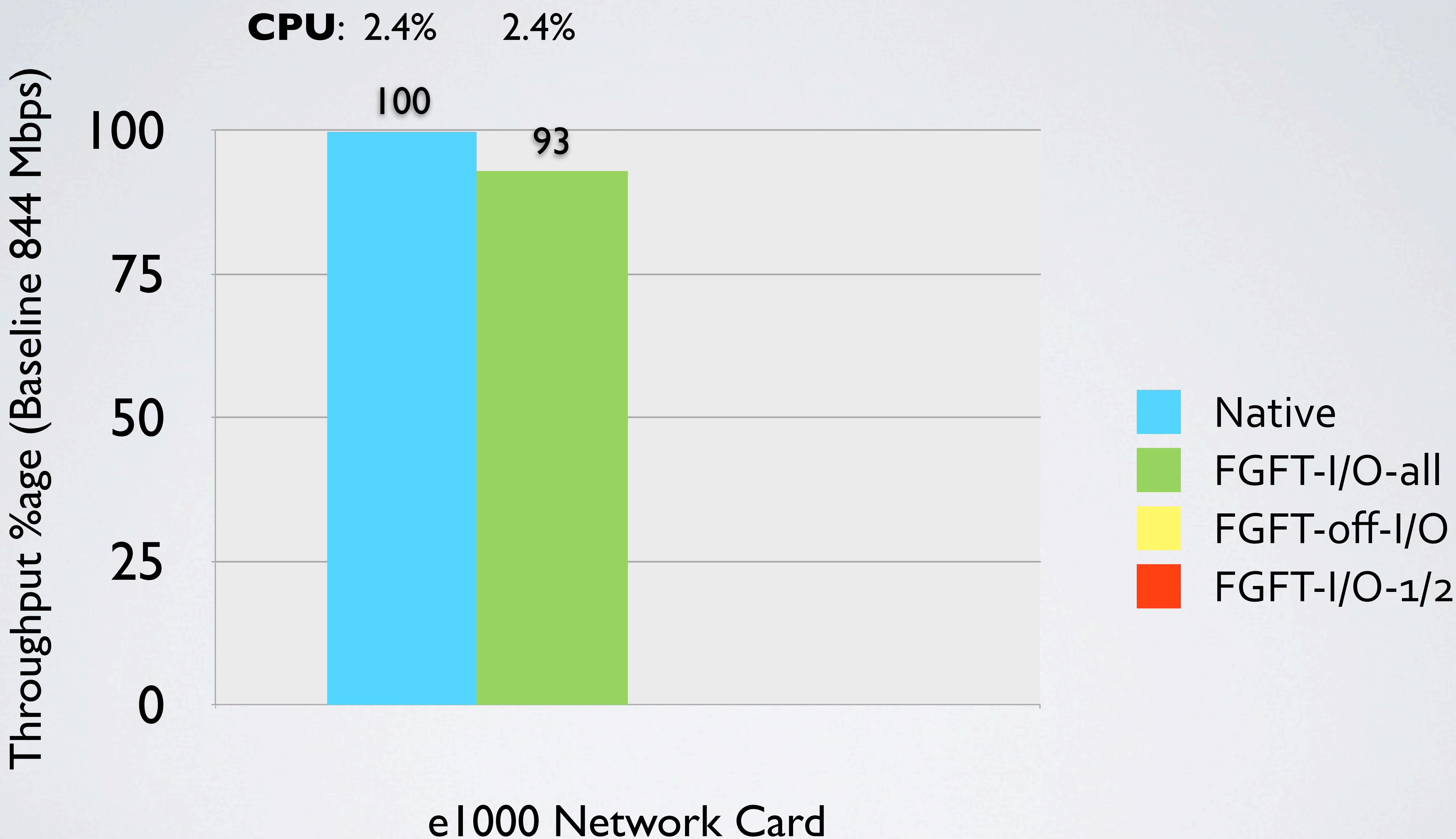
netperf on Intel quad-core machines

Throughput with isolation and recovery



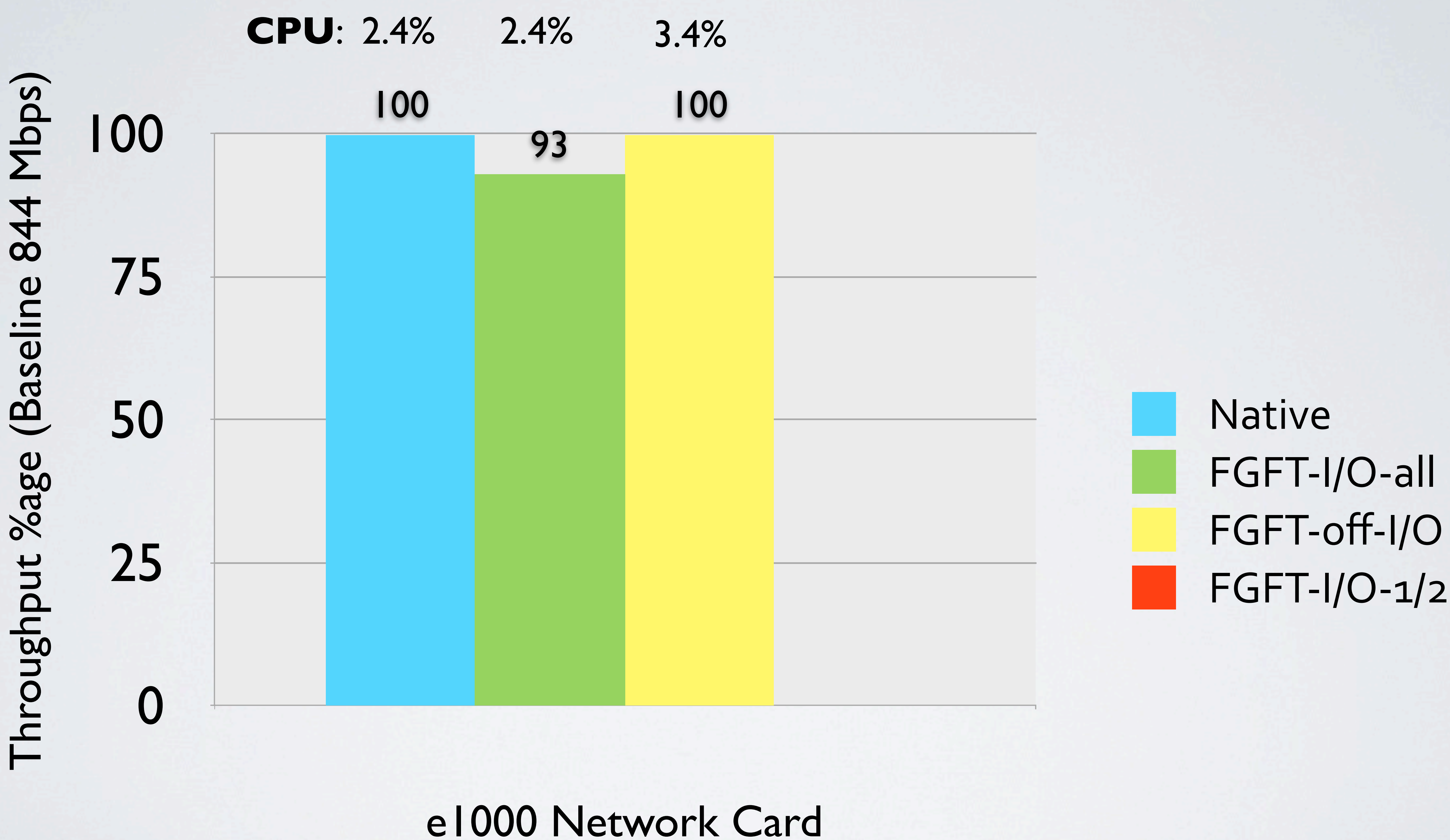
netperf on Intel quad-core machines

Throughput with isolation and recovery



netperf on Intel quad-core machines

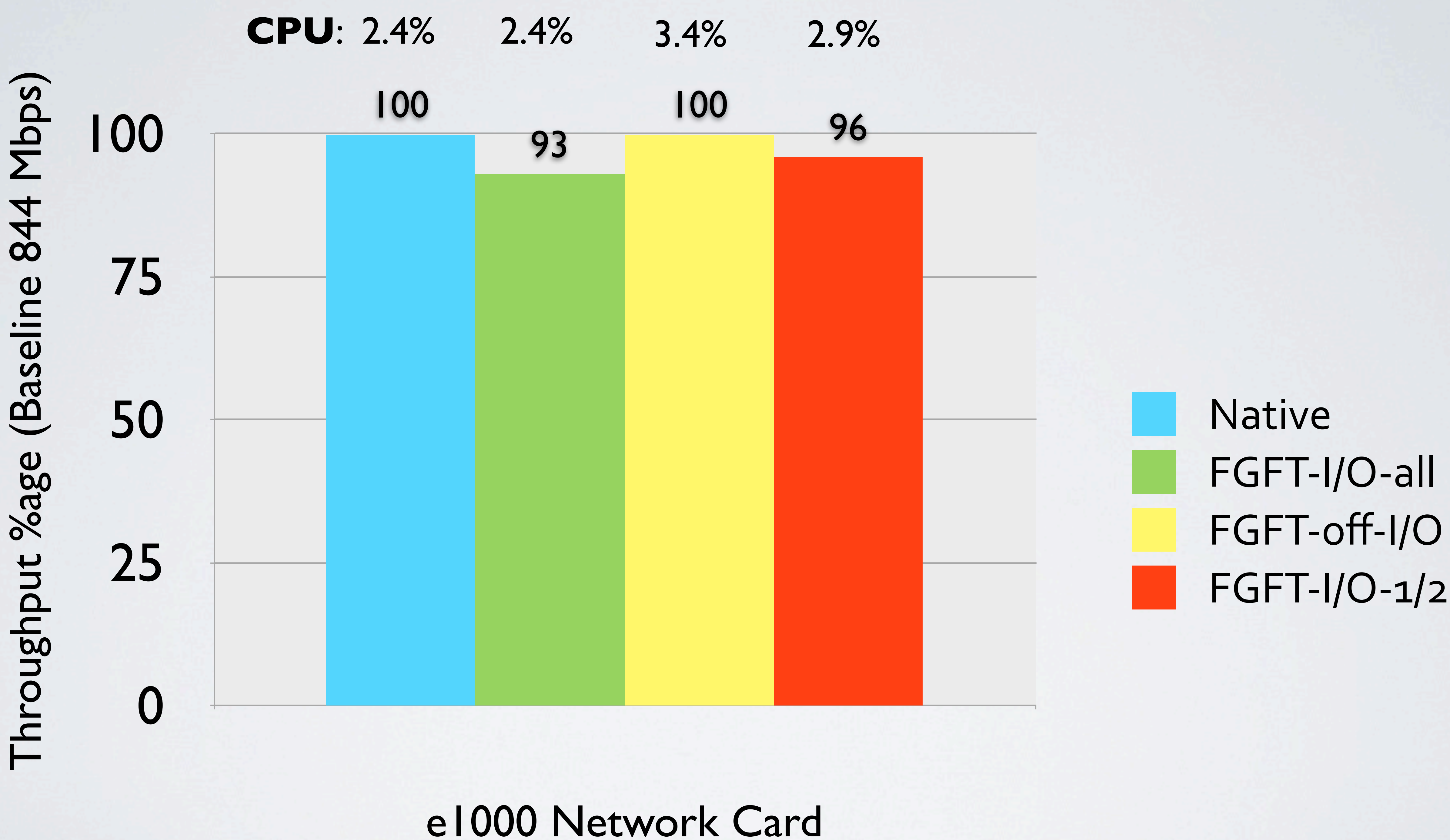
Throughput with isolation and recovery



netperf on Intel quad-core machines

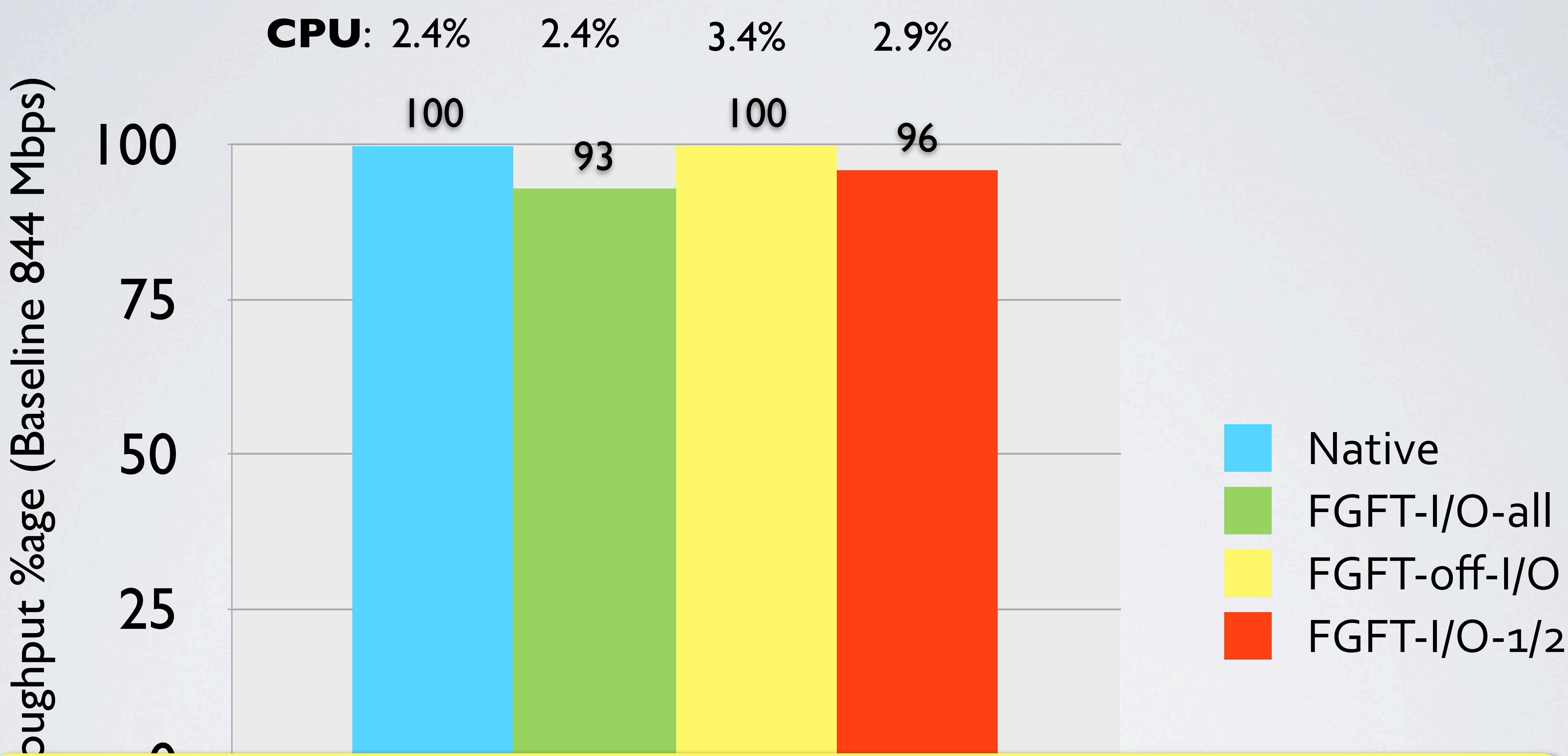
25

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

- FGFT runs driver code as transactions
  - Provides fault tolerance at incremental performance and programmer efforts
- Introduced device checkpoints
  - Provides fast and complete recovery semantics
- Fast device checkpoints should be explored in other domains like fast reboot, upgrade etc.

Questions

Asim Kadav

- **http://cs.wisc.edu/~kadav**
- **kadav@cs.wisc.edu**
- **Graduating in spring!**

Extra slides

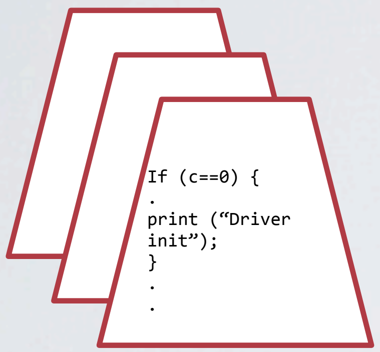
- Unlike suspend, devices continue to be accessed after a checkpoint
  - Rely on drivers following ACPI specifications for correctness

Latency for device checkpoint/restore

Driver	Class	Bus	Checkpoint Times	Restore Times
8139too	net	PCI	33µs	62µs
e1000	net	PCI	32µs	280ms
r8169	net	PCI	26µs	30µs
pegasus	net	USB	0µs	4ms
ens1371	sound	PCI	33µs	111ms
psmouse	input	serio	0µs	390ms

Fast checkpoint/restore using suspend/resume

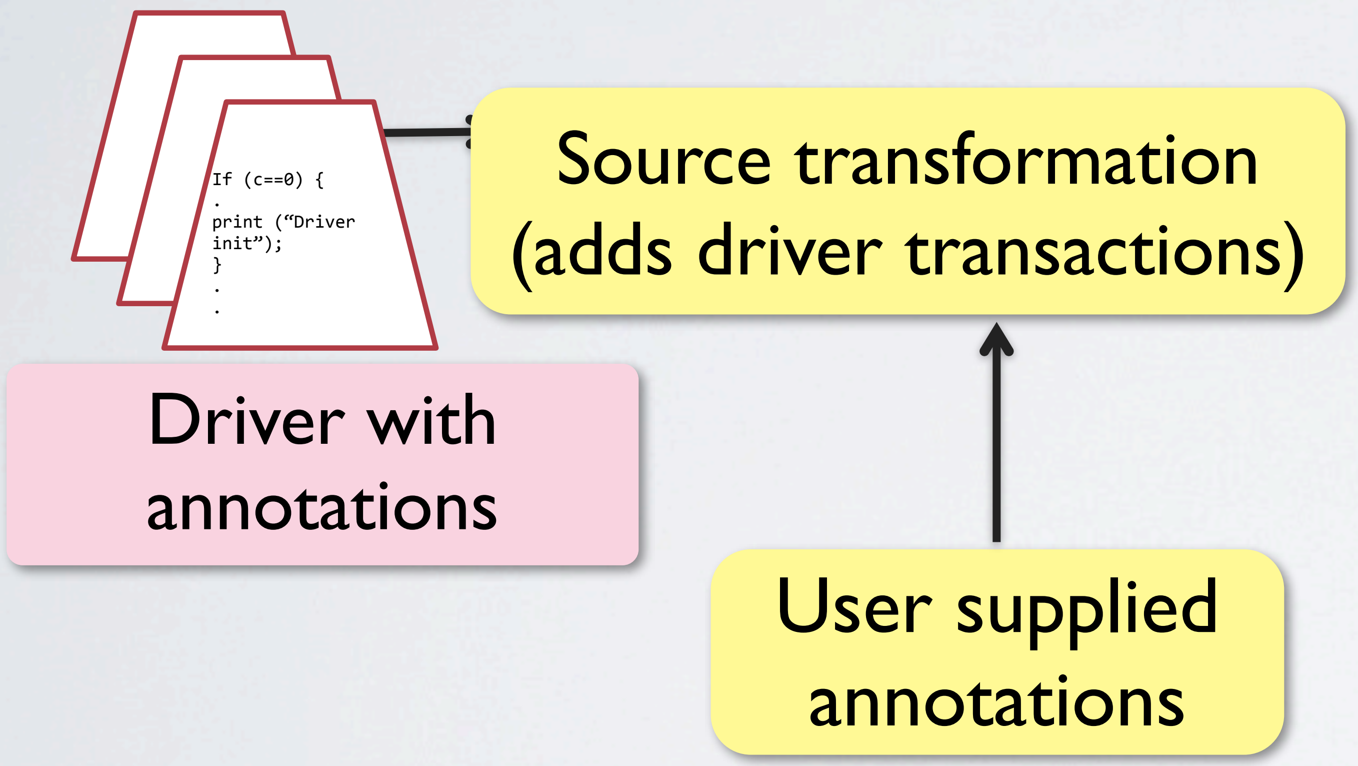
Transforming drivers to run as FGFT



Driver with annotations

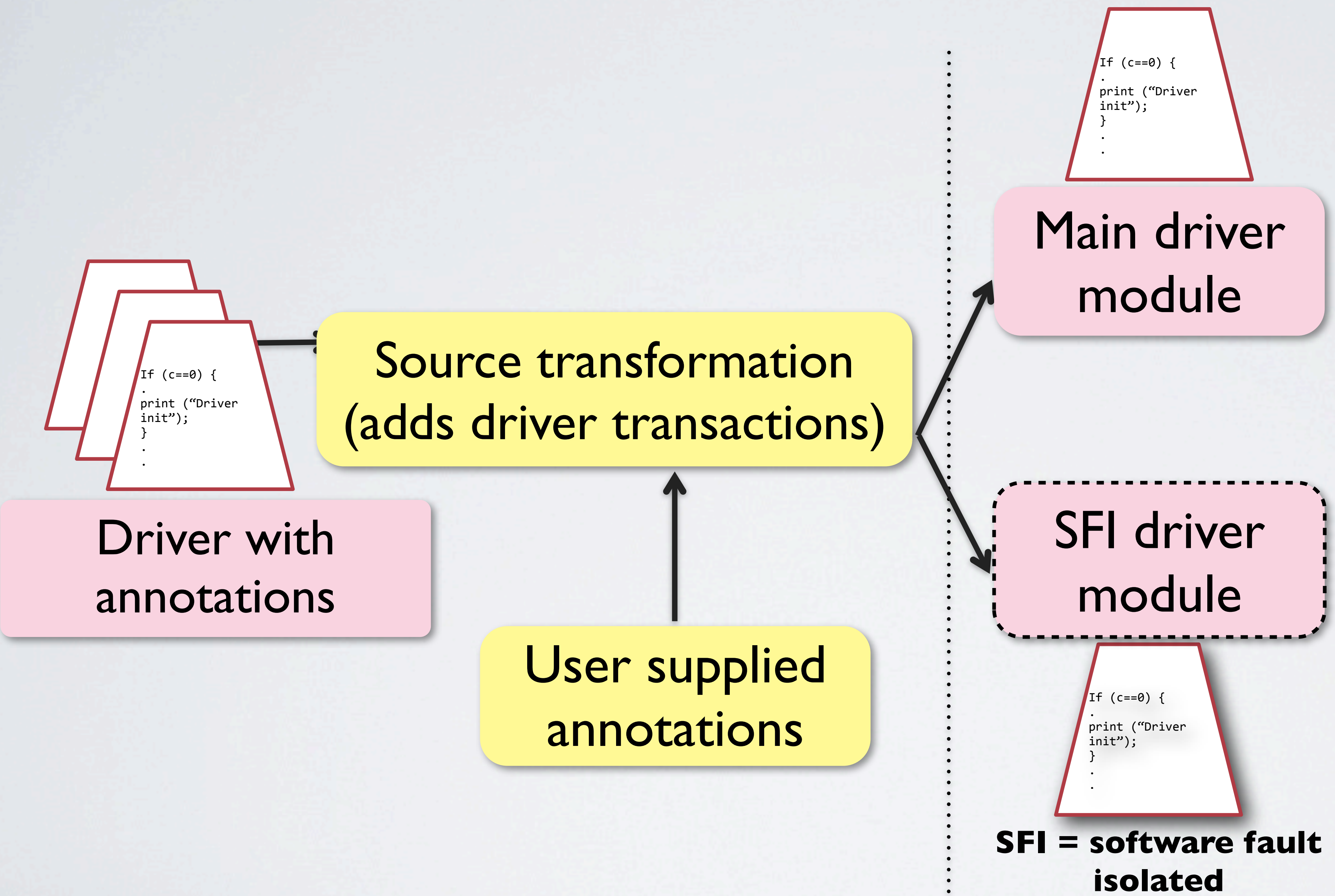
Static modifications

Transforming drivers to run as FGFT



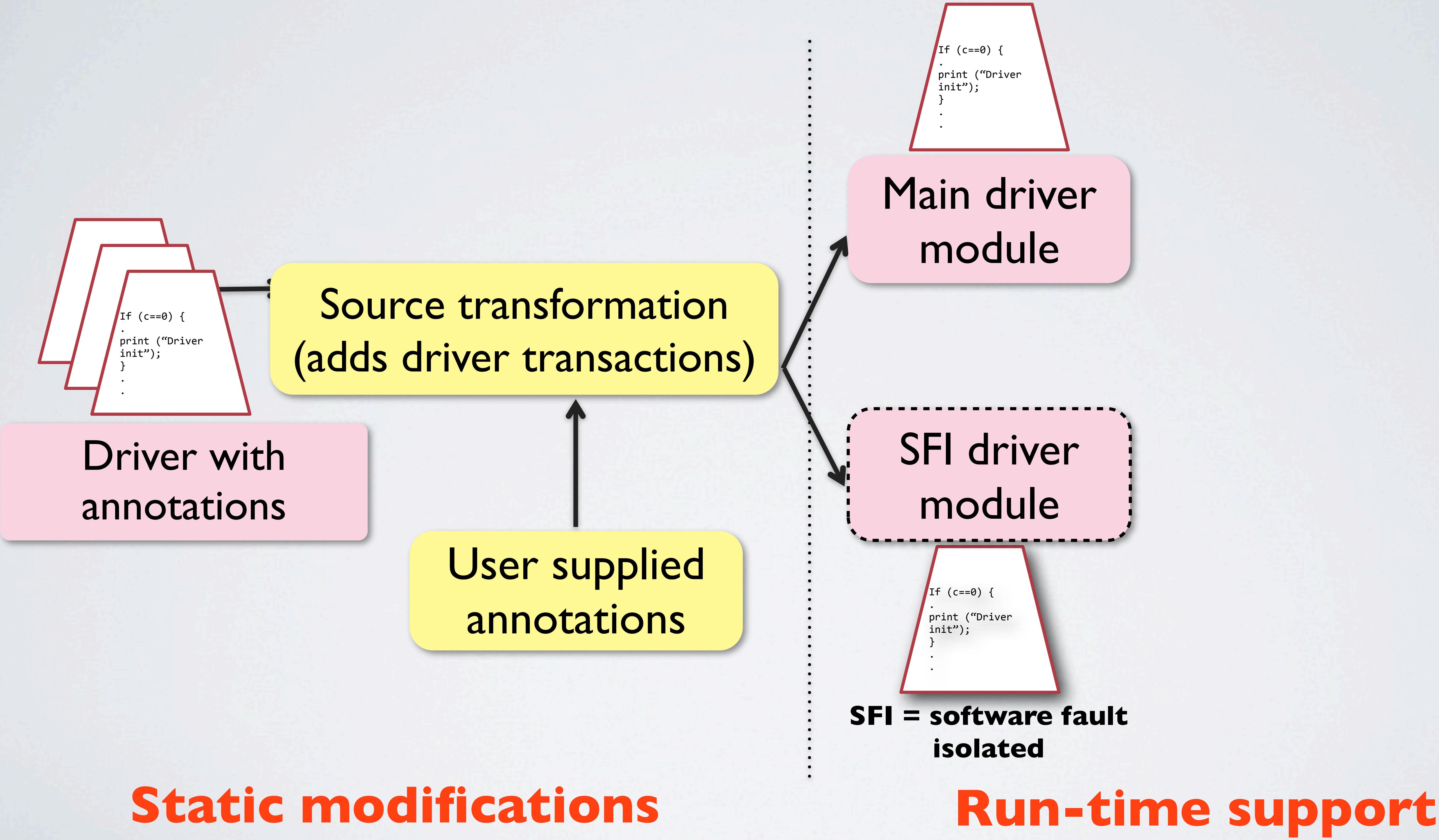
Static modifications

Transforming drivers to run as FGFT



Static modifications

Transforming drivers to run as FGFT



Transforming drivers to run as FGFT