IEEE/IFIP International Conference
on Dependable Systems and
Networks
Florence, Italy, June 28-July 1, 2004

Characterization of the Impact of Faulty Drivers on the Robustness of the Linux Kernel

Arnaud Albinet, Jean Arlat, Jean-Charles Fabre

The European Commission
Community Research
ist
information society
technologies

DBench
DependabilityBenchmarking

Outline

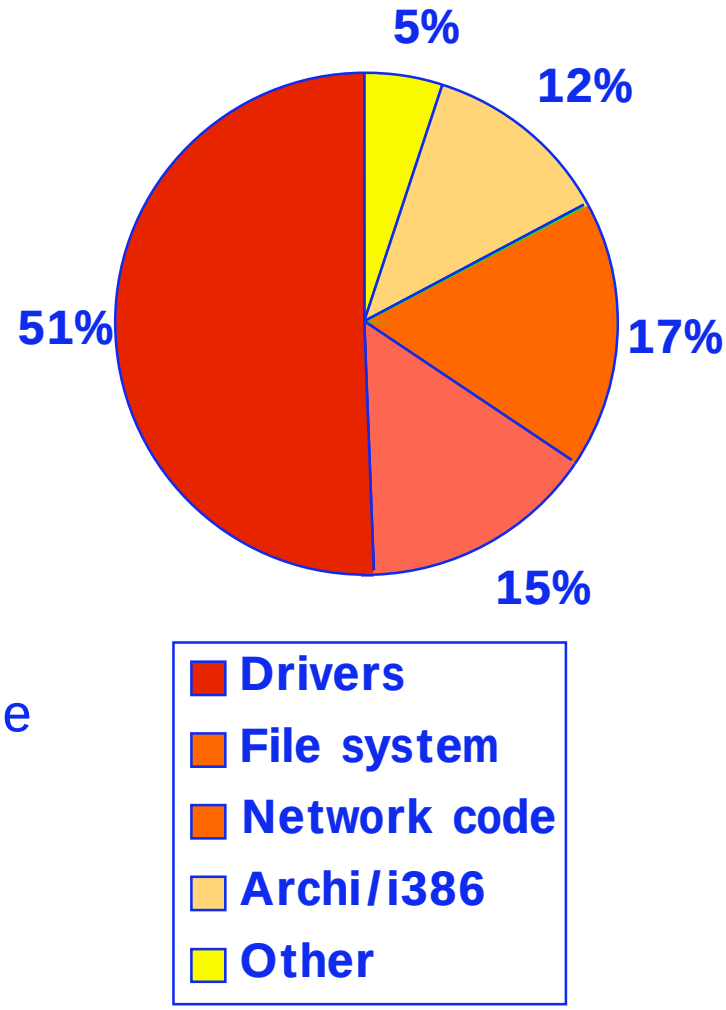
- Motivation, Context and Objectives
- The Approach and Testbed
- Examples of Results and Analyses
- Conclusion and On-Going Work

Motivation

- **Drivers account for a large proportion of reported OSs failures**

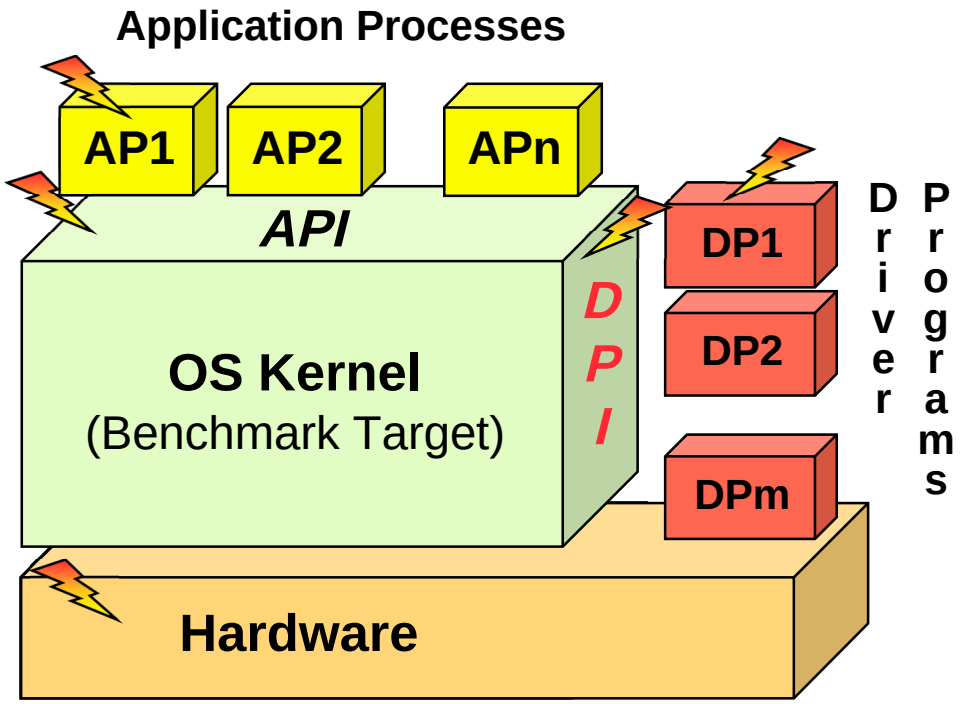
- **Error distribution (Linux)**
  [Chou *et al.* 2001]

- **Main Rationale**
  - Developpers are “outsiders”
  - Drivers form an increasing part of OS code (already 70% of LOC in Linux 2.4.1)



Context and Objectives

- **OS Kernel Interfaces:**
  - Hardware Layer
  - Application Processes
  - Driver Programs
- **Possible approaches targeting drivers:**
  - Code mutation
  - Interface Driver-Kernel



- Definition of a Driver Programming Interface
- Linux as a Target
- Framework for Accommodating Various Dependability Concerns

The Proposal for the *Linux* DPI

- **The drivers make use of specific system calls to perform tasks. These are denoted symbols (functions, constants & variables) in the case of *Linux*** (more than 1000 symbols in release 2.4.18 including ≈ 700 kernel functions)

Categories	Typical Symbols
Interrupt Management	Kmalloc, kfree, free_pages, exit_mm ,...
File System Management	add_timer, del_timer, request_irq, free_irq, irq_stat, add_wait_queue, finish_wait, ...
Control Block Management	fput, fget, iput, follow_up, follow_down, filemap_fdatawrite, filemap_fdatawait, lock_page, ...
Others	…

- **Example `int request_irq`**
  (allocation of a peripheral device to an interrupt channel)

```
int request_irq(unsigned int irq, void (*handler)(),
unsigned long irqflags, const char * devname, void *dev_id)
```

Channel allocated

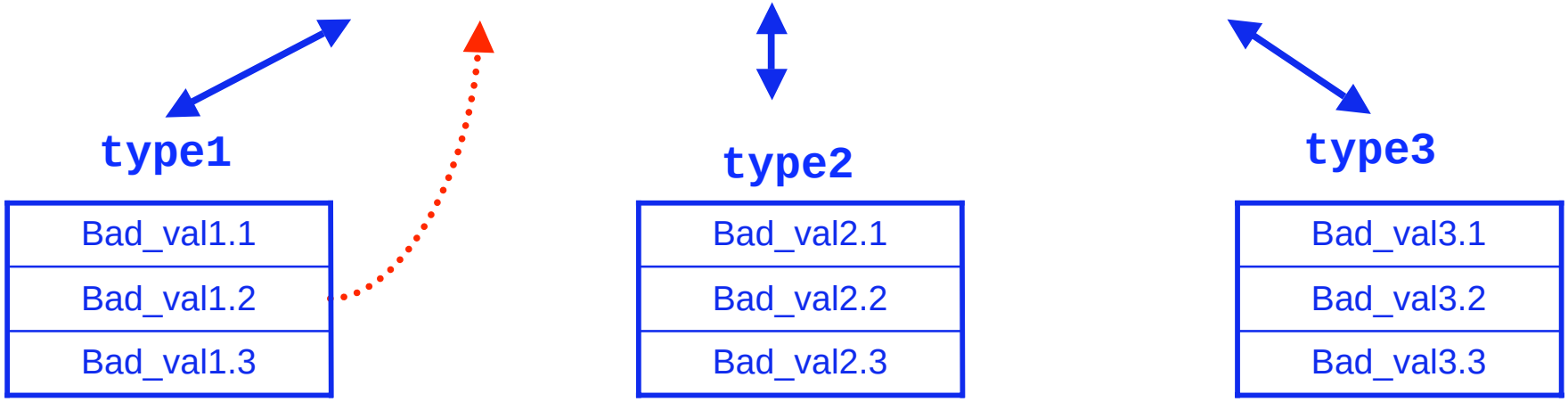
pointer to the interrupt manager

Etc. ...

Robustness Testing at the DPI Level

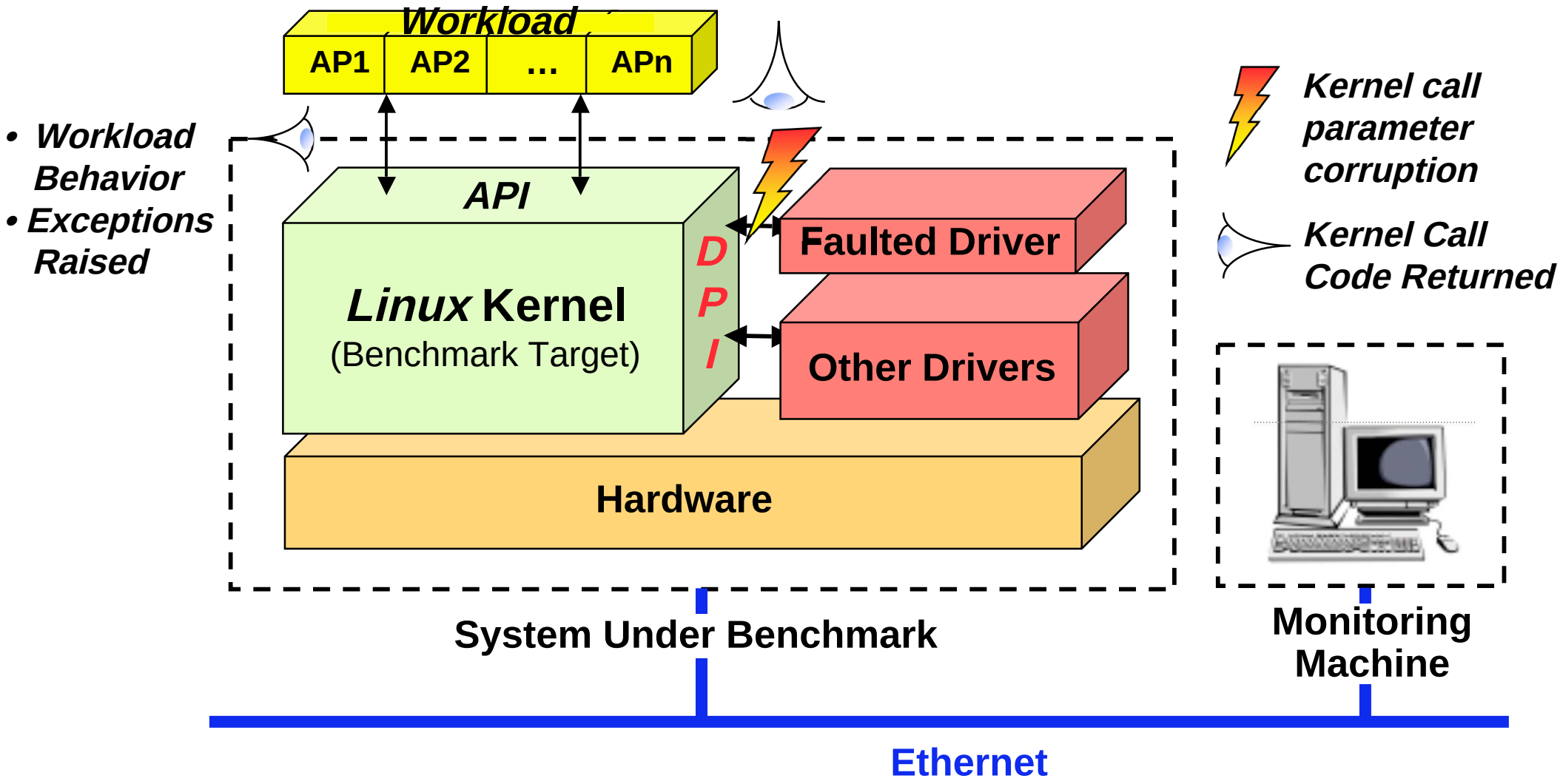
- **SWIFI -> Corruption of kernel call parameters**

```
void foo (type1 par1,   type2 par2,   type3 par3, …);
```



Type	Corrupted value 1	Corrupted value 2	Corrupted value 3
int	INT_MIN(0x80000000)	0	INT_MAX(0x7FFFFFFF)
unsigned int	0	INT_MIN(0x80000000)	ULONG_MAX(0xFFFFFFFF)
unsigned short	0	SHRT_MIN (0x8000)	USHRT_MAX(0xFFFF)
* (pointer)	NULL	random( )	All bits = 1 (0xFFFFFFFF)

Experimental Context: The Testbed



Experimental Context: Considered Drivers and Workload

- **Benchmark Target and System Under Benchmark**
  - *Linux* Kernel 2.2.20 et 2.4.18
  - Distribution Debian 3.0
  - Hardware architecture x86 Pentium
- **Target Drivers**
  - Network Card drivers (SMC-ultra, Ne2000)
  - Sound Card driver (Soundblaster)
  - ...
- **Workload**
  - Several specific workloads dedicated to each of the drivers

De-installation — Re-installation
— Series of Requests —
De-installation — Re-installation

Experimental Context: Main Outcomes

- **Internal**

  Error code (EC) returned to the driver at the level of the DPI

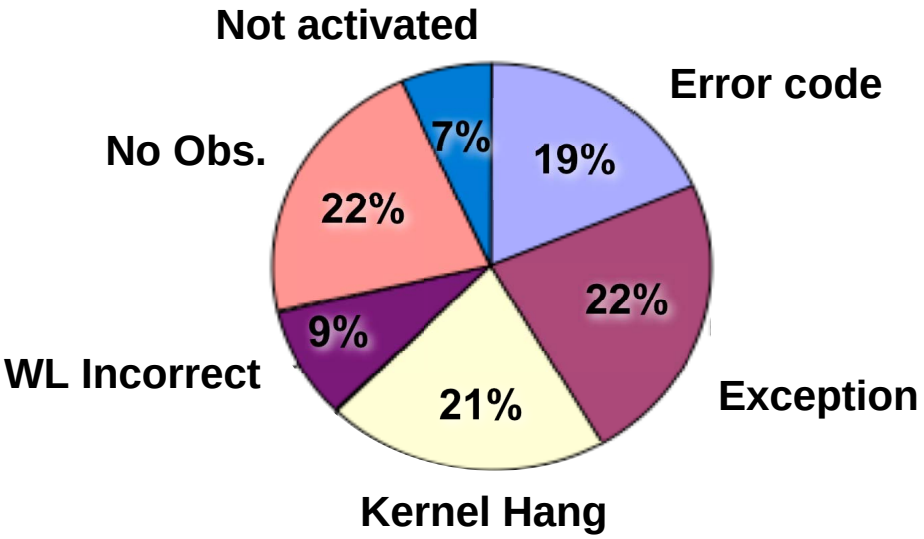
- **External**
  - Exception (XC): Events raised by the hardware & the Kernel and observable at the level of the API
  - Kernel Hang (KH): The Kernel does not reply
  - Workload Aborted (WA): The workload is abruptely interrupted (some service requests could not be made)
  - Workload Incorrect (WI): The workload has completed, but has failed to execute correctly all the service
  - Workload Completed (WC): In order to measure the duration of the workload

Possible Outcomes and Diagnoses

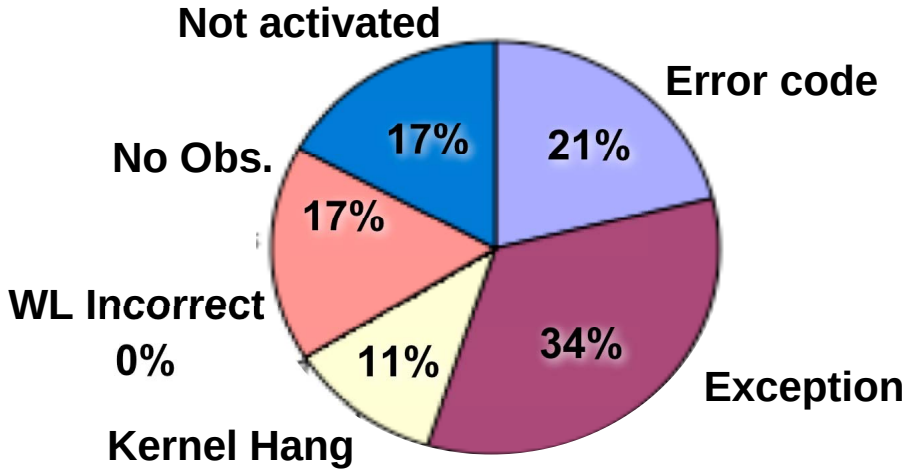
Outcomes					
#	Notification		Failure Modes		
	EC	XC	WA	WI	KH
O1	1	0	0	0	0
O2	1	1	0	0	0
O3	0	1	0	0	0
O4	1	1	0	0	1
O5	1	0	0	0	1
O6	0	1	0	0	1
O7	0	0	0	0	1
O8	1	1	1	X	1
O9	1	0	1	X	1
O10	0	1	1	X	1
O11	0	0	1	X	1
O12	0	0	0	0	0
O13	1	1	1	X	0

Some Results - Network Card Drivers (first event)

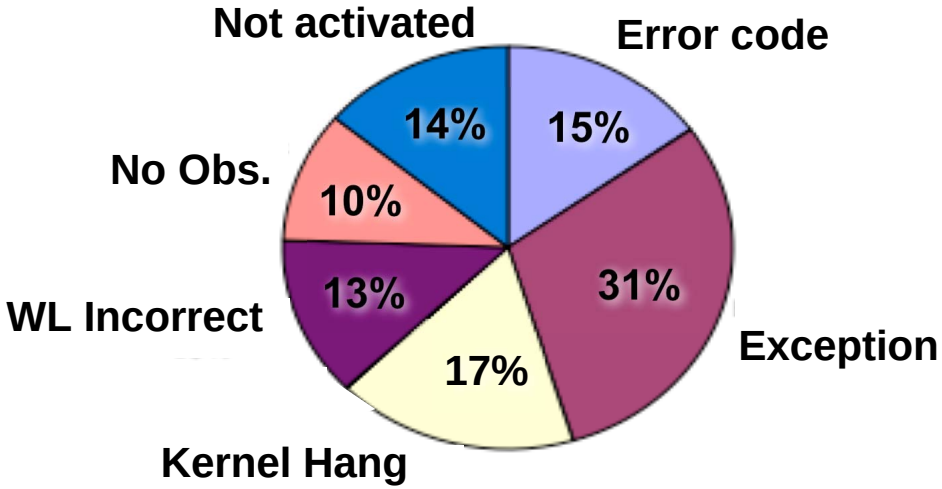
SMC-ultra Driver — Linux V2.2



SMC-ultra Driver — Linux V2.4



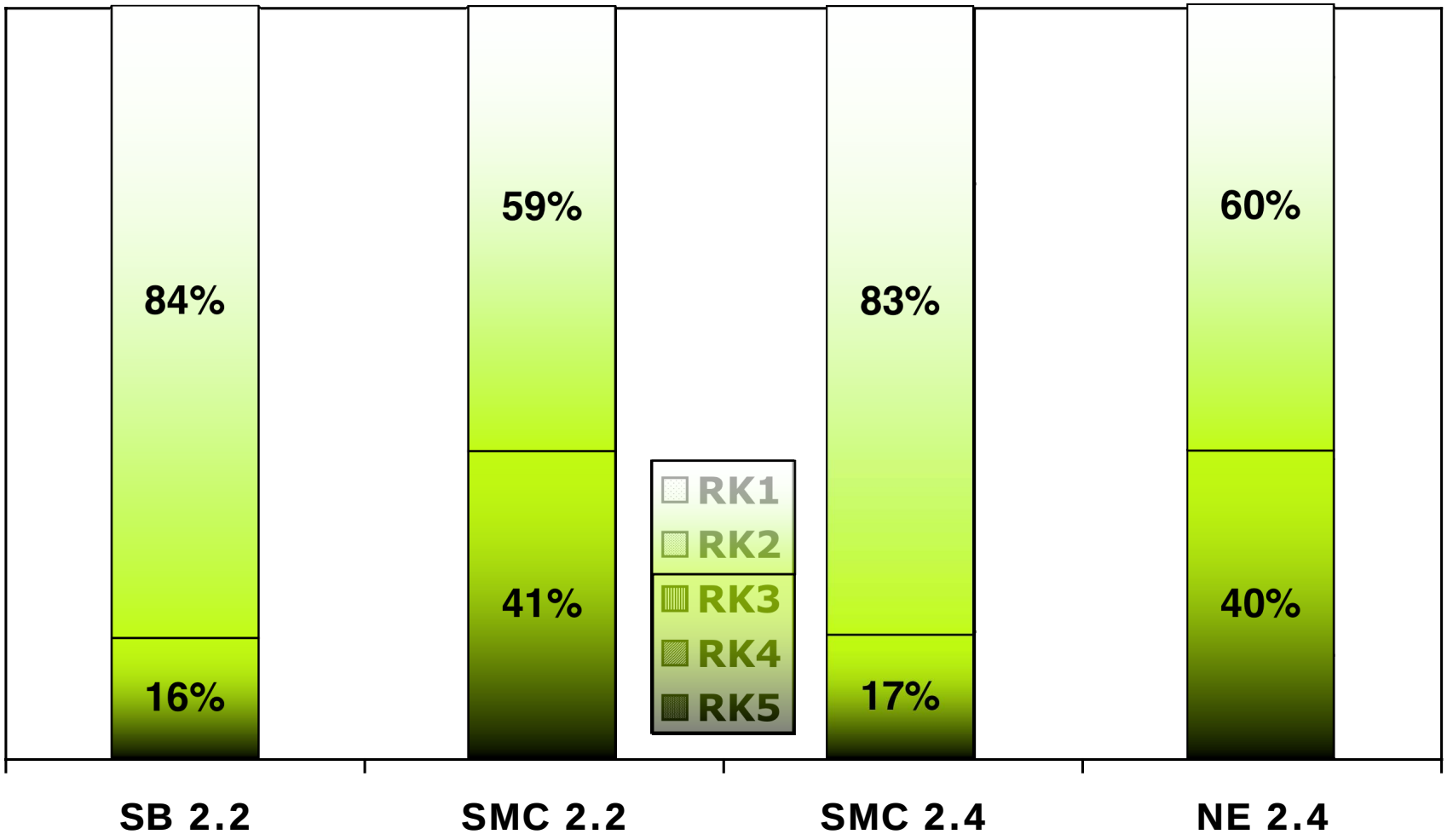
NE2000 Driver — Linux v2.4



Measures and Viewpoints

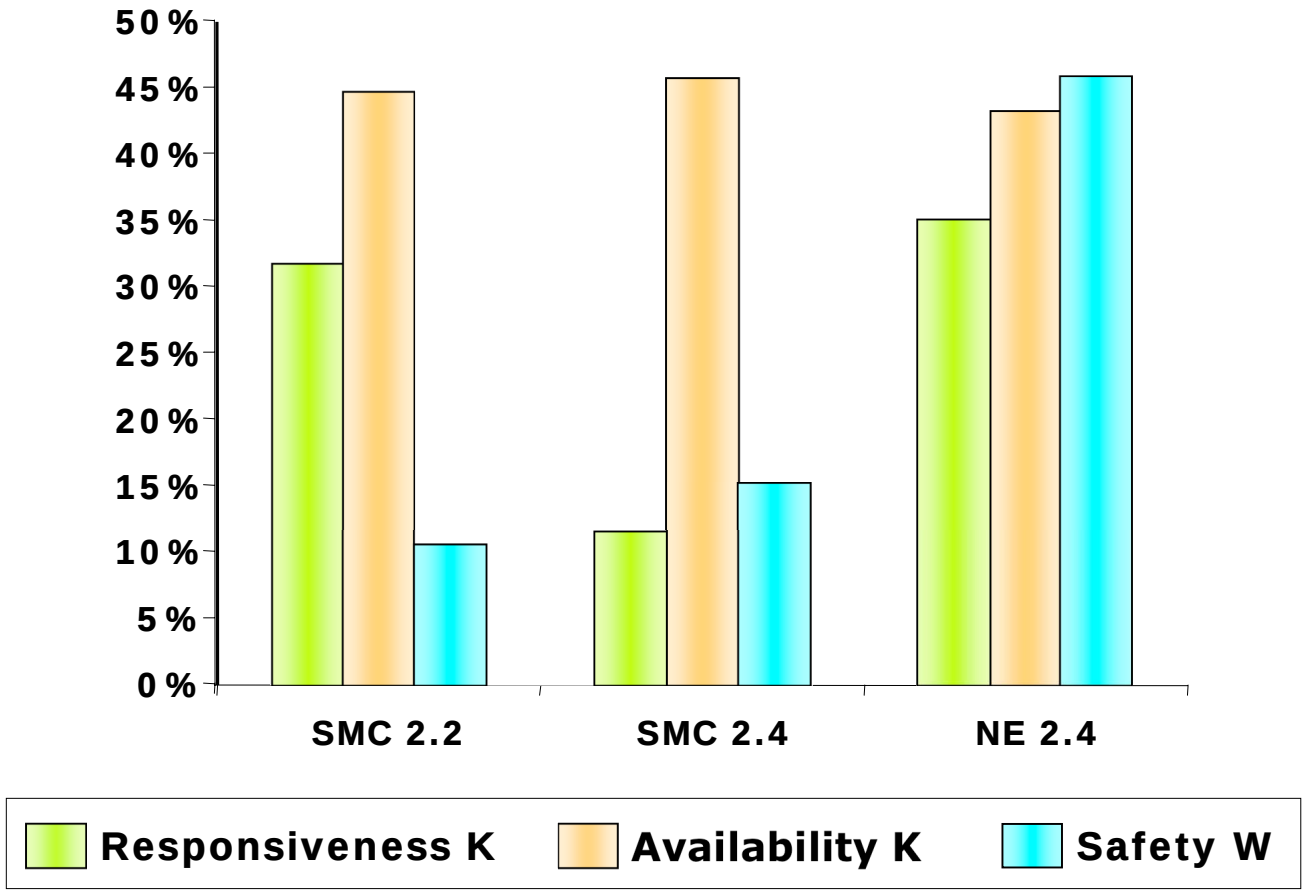
	ID	All Outcomes (exc. O12)	Description
Responsiveness of the Kernel	RK1	O1-O3	An error is notified by the kernel before the WL completes correctly
	RK2	O4-O6, O8-O10, O13-O15, O21-O23	An error is notified by the kernel before a failure is observed
	RK3	O16	No error is notified and the WL is aborted
	RK4	O7, O11,O24	No error is notified and the Kernel hangs
	RK5	O20	No error is notified and the WL completes incorrectly
Availability of the Kernel	AK1	O1-O3	The WL completes correctly and an error is notified by the Kernel
	AK2	O13-O20	The WL is aborted or completes incorrectly
	AK3	O4-O7	The WL hangs or the WL completes correctly
	Ak4	O8-O11, O21-O24	The WL hangs or the WL is aborted or completes incorrectly
Safety of the WL	SW1	O1-O3	The WL completes correctly and an error is notified by the Kernel
	SW2	O6-O7	The WL completes correctly and the Kernel hangs
	SW3	O8-O11, O13-O16	The WL is aborted or the Kernel hangs
	SW4	O13-O16	The WL completes incorrectly and the Kernel hangs
	SW5	O17-O20	The WL completes incorrectly and the Kernel does not hang

Kernel Responsiveness Viewpoint



Analyses According To Different Viewpoints
Example: Network Card Drivers

”The considered property is not satisfied”



Concluding Remarks

- Specific and suitable approach to develop robustness benchmarks wrt driver errors
- Analysis framework that accommodates various dependability concerns
- Current experiments: Assess the portability to other Kernels (Windows, MacOS X)
- Complement the scope of a *Dependability Benchmark* aimed at characterizing the robustness of an OS