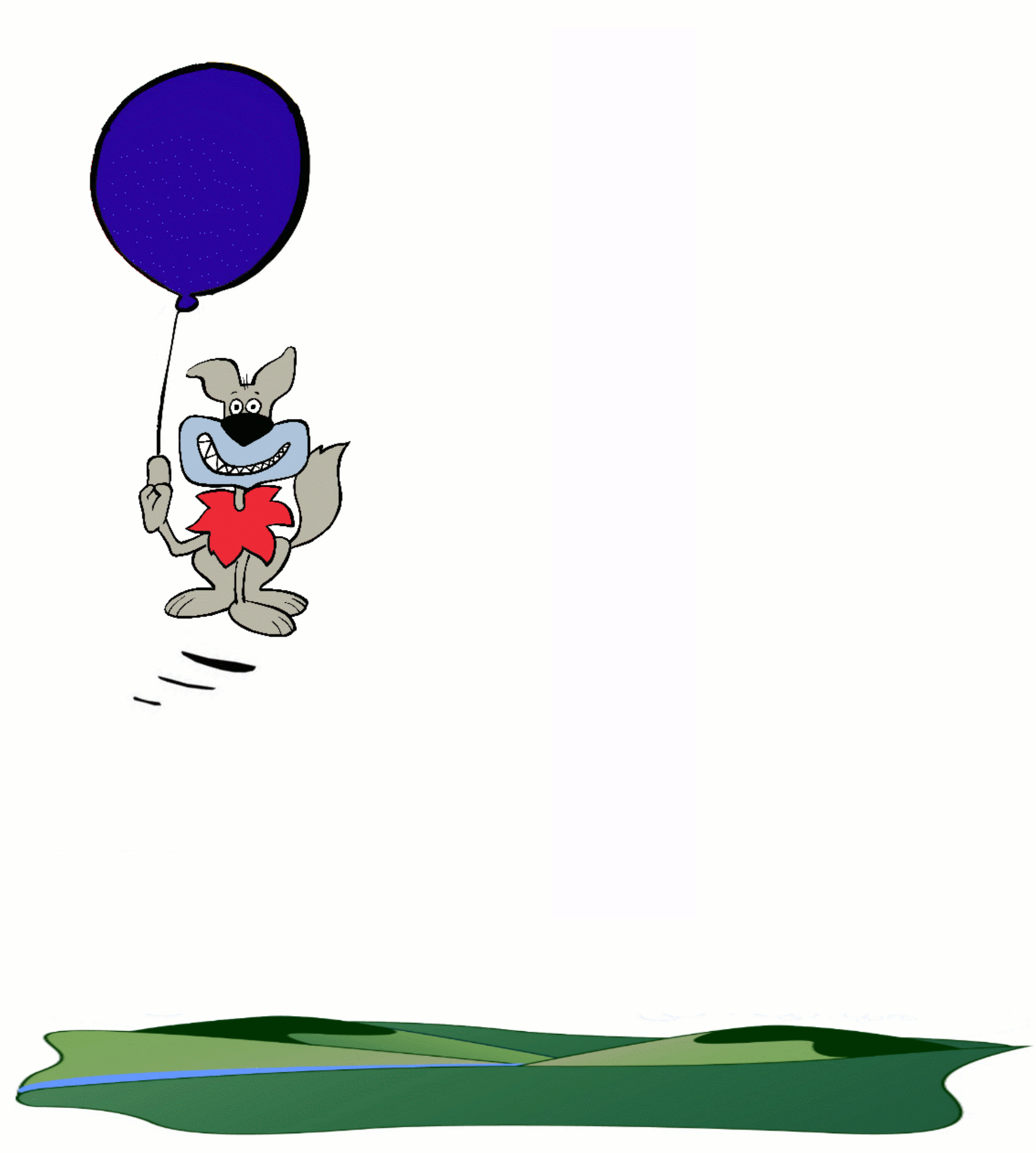
CONSTRUCTION OF A HIGHLY DEPENDABLE OPERATING SYSTEM

6th European Dependable Computing Conference

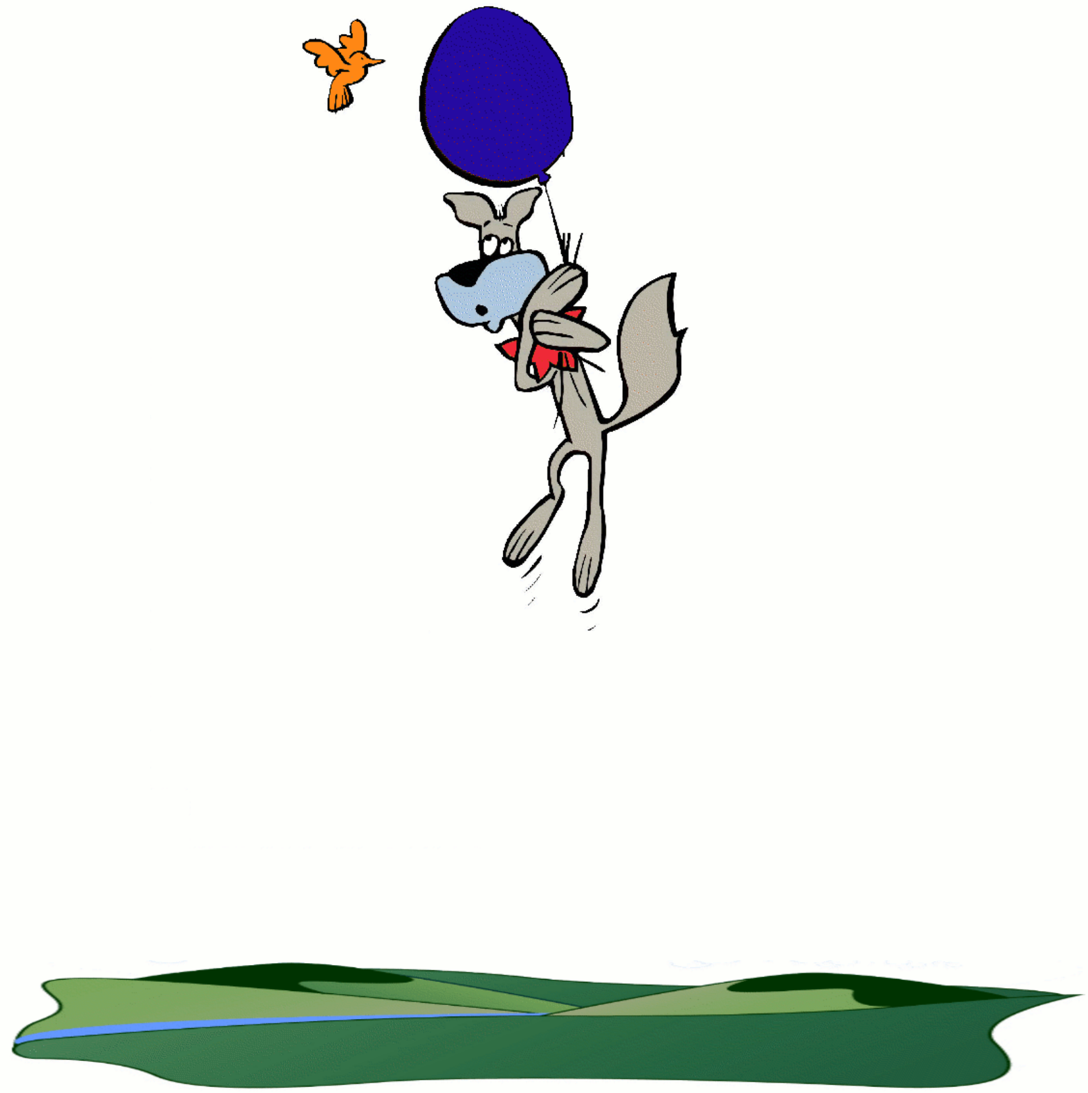
Coimbra, Portugal
October 18-20, 2006

Jorrit N. Herder
Dept. of Computer Science
Vrije Universiteit Amsterdam

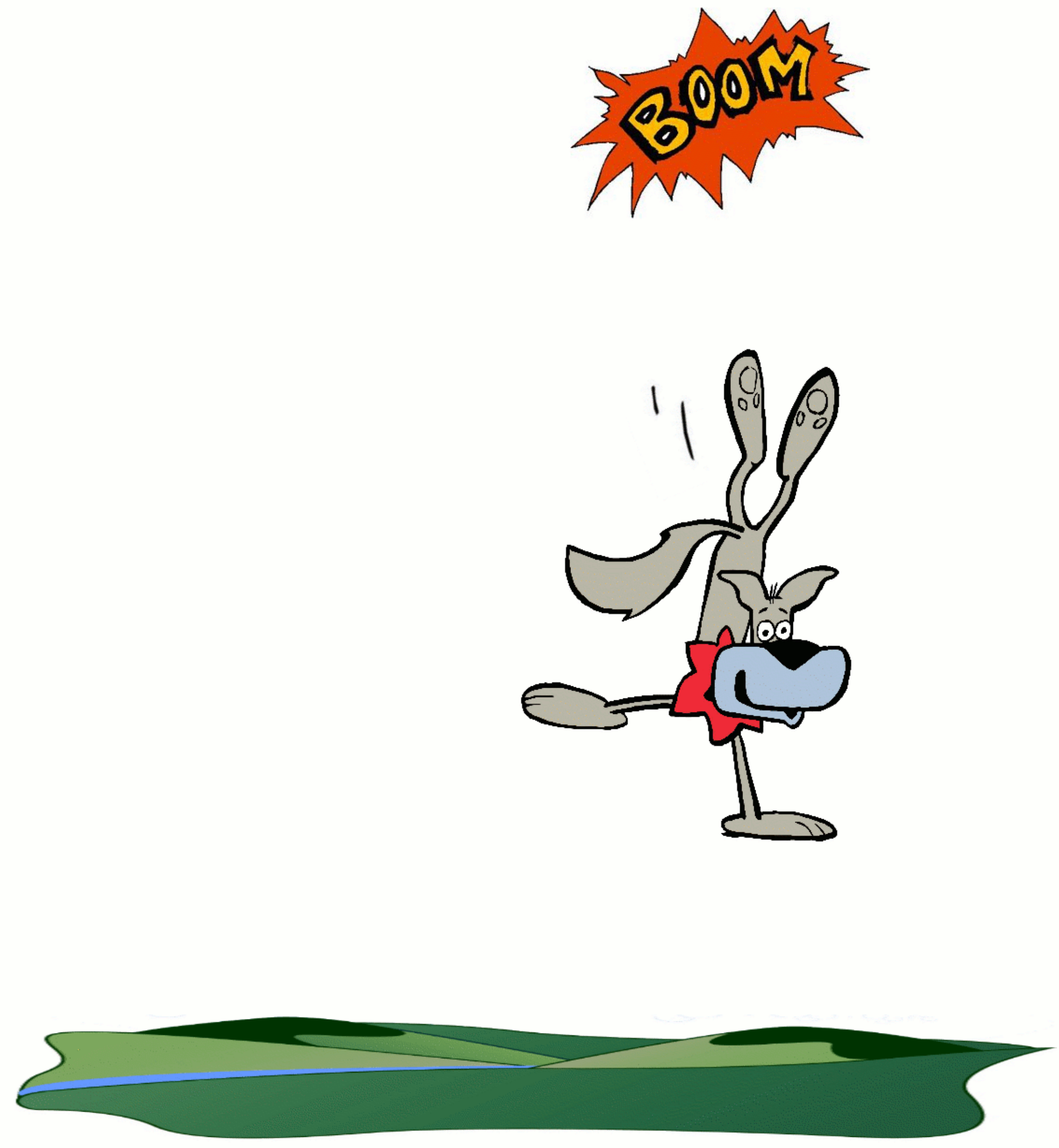
THE "BALLOON" OPERATING SYSTEM



THE “BALLOON” OPERATING SYSTEM



THE “BALLOON” OPERATING SYSTEM

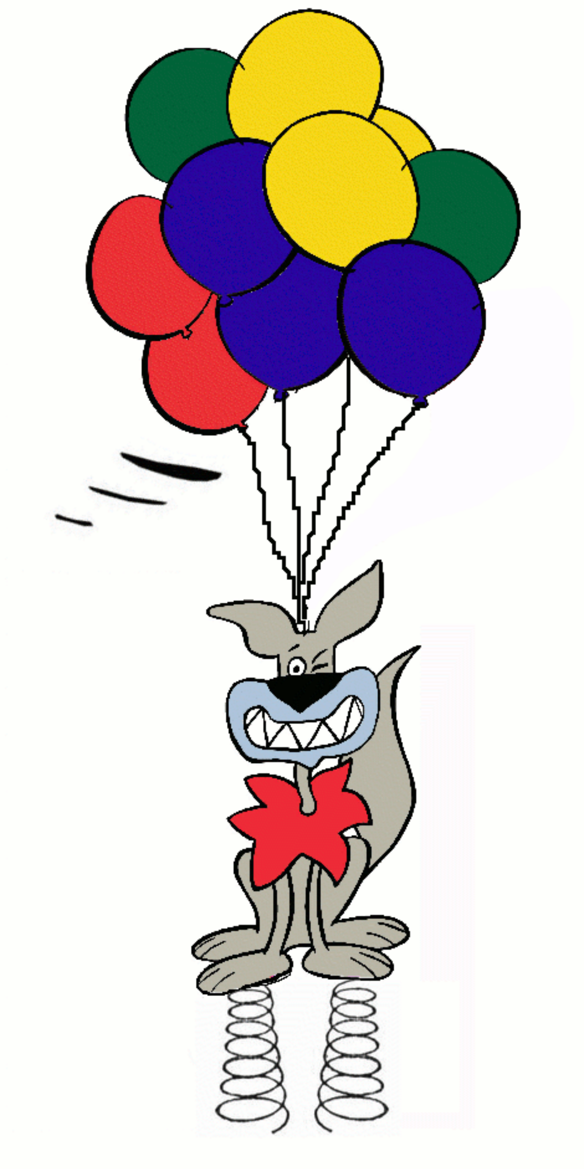


THE “BALLOON” OPERATING SYSTEM

IMPROVING OPERATING SYSTEM DEPENDABILITY

MINIX 3: a highly dependable OS

- Single failure no longer fatal
- Spring back after failure



TALK OUTLINE

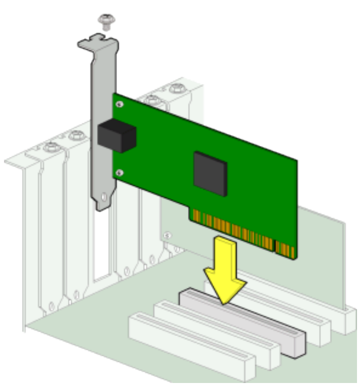
- Welcome (done)
- Problem statement (next)
- Construction work
- Dependability features
- Performance statistics
- Discussion and conclusion

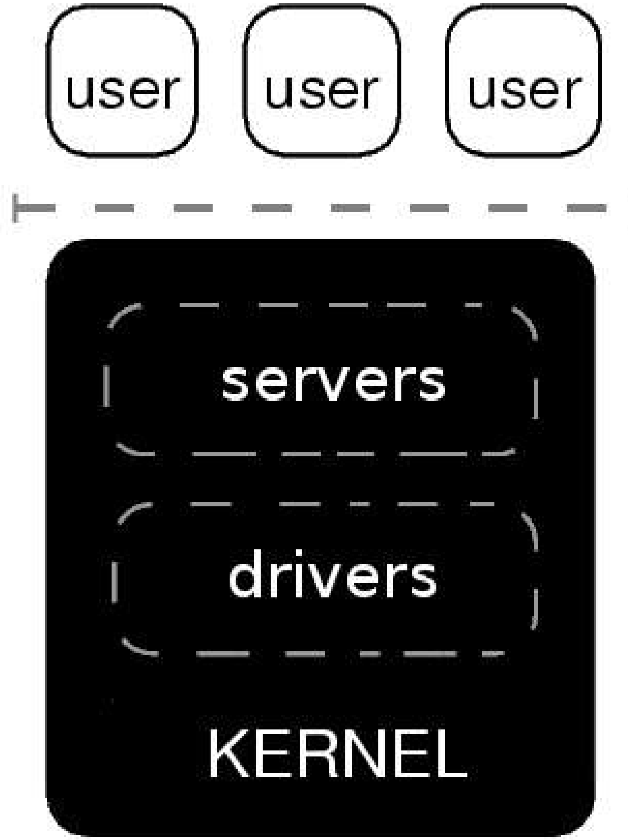
INTRODUCTION

Jorrit N. Herder <jnherder@cs.vu.nl>

DRIVERS IN A MONOLITHIC OPERATING SYSTEM

- **Device drivers control hardware**

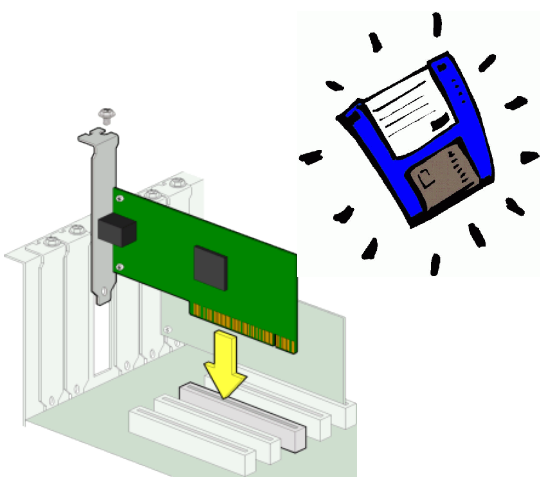


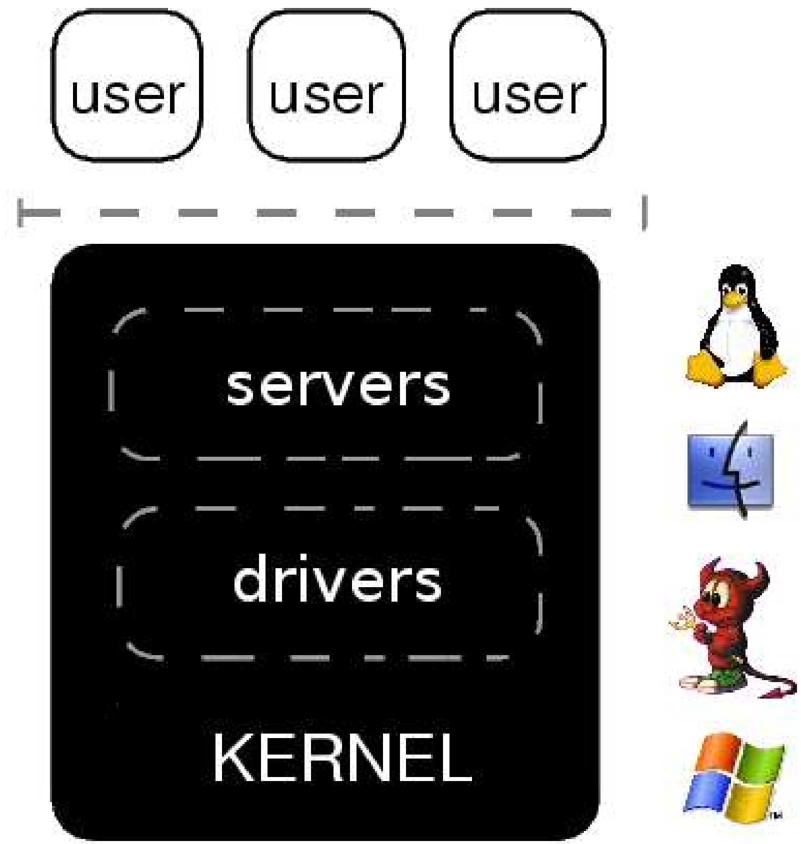


Jorrit N. Herder <jnherder@cs.vu.nl>

DRIVERS IN A MONOLITHIC OPERATING SYSTEM

- Device drivers control hardware
- Driver is run within the kernel

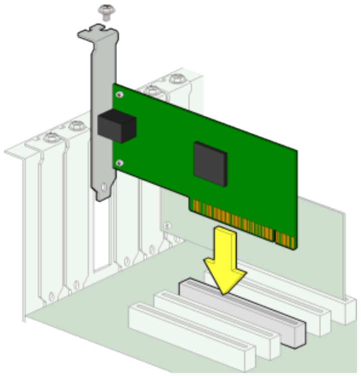


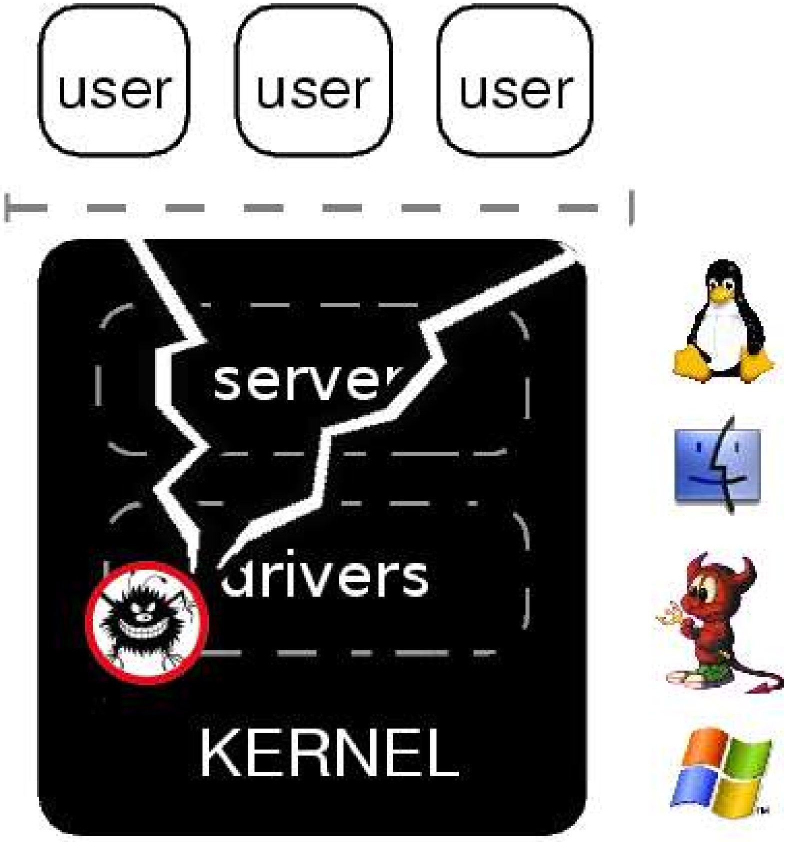


Jorrit N. Herder <jnherder@cs.vu.nl>

DRIVERS IN A MONOLITHIC OPERATING SYSTEM

- Device drivers control hardware
- Driver is run within the kernel
- Bugs can easily spread



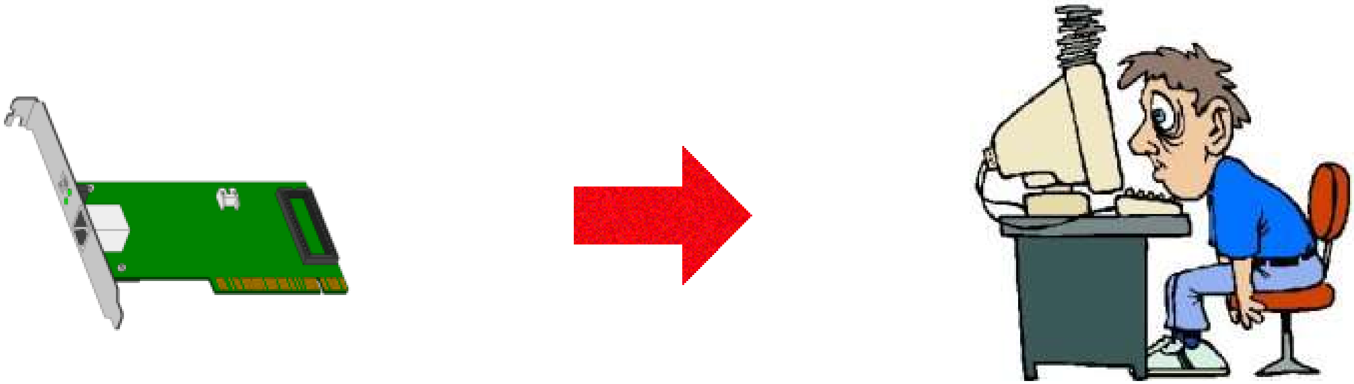


INHERENT PROBLEMS OF MONOLITHIC DESIGNS

- **Fundamental design flaws in monolithic kernels**
  - All code runs at highest privilege level (breaches POLA)
  - No proper fault isolation (any bug can be fatal)
  - Huge amount of code *in* kernel (6-16 bugs per 1000 LOC)
  - Untrusted, 3rd party code in kernel (70% of code, more bugs)
  - Entangled code increases complexity (hard to maintain)

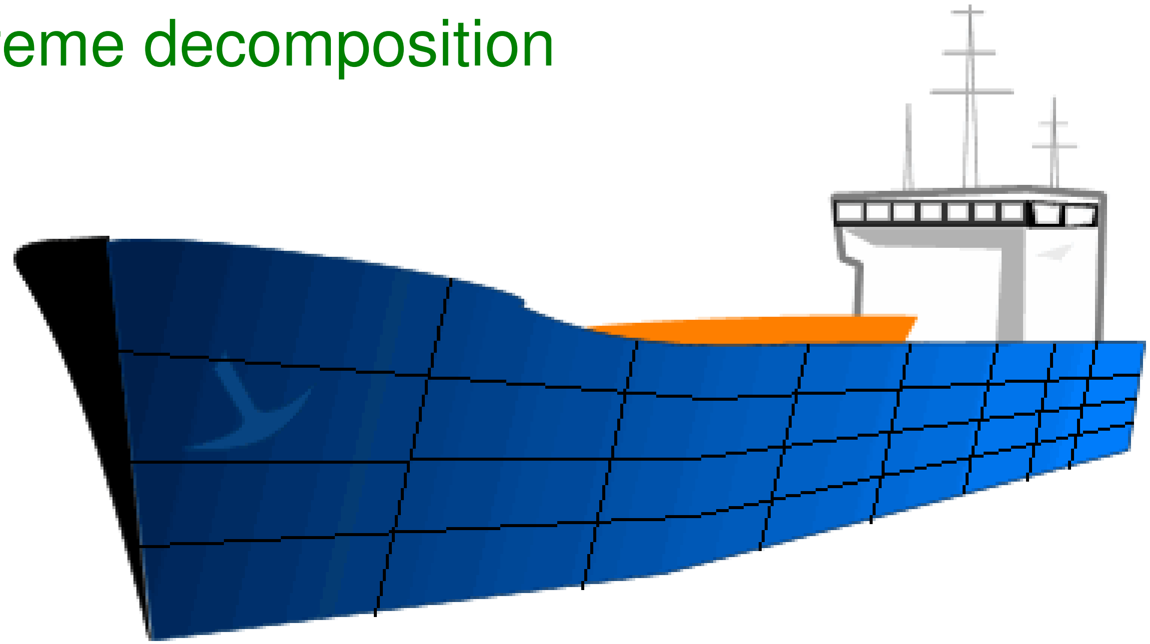
INHERENT PROBLEMS OF MONOLITHIC DESIGNS

- **Fundamental design flaws in monolithic kernels**
  - All code runs at highest privilege level (breaches POLA)
  - No proper fault isolation (any bug can be fatal)
  - Huge amount of code *in* kernel (6-16 bugs per 1000 LOC)
  - Untrusted, 3rd party code in kernel (70% of code, more bugs)
  - Entangled code increases complexity (hard to maintain)



HOW ABOUT MODULAR DESIGNS?

- **Modularity is commonly used in other engineering disciplines**
  - Ship's hull is compartmentalized to improve 'dependability'
  - Aircraft carrier is build out of many, well-isolated parts
- **Use modularity to improve OS dependability**
  - We propose an extreme decomposition



CONSTRUCTION

UNDERLYING IDEA

“Perfection is not achieved when there is nothing left to add, but when there is nothing left to take away.”

-- Antoine de Saint-Exupéry

MINIX 3: A HIGHLY RELIABLE OPERATING SYSTEM

- **Microkernel design (< 4000 LOC)**
  - Low-level operations to support user-space OS
- **OS runs as set of isolated user-mode servers and drivers**
  - MMU protection and various other encapsulation properties
- **Mechanisms to detect and repair failures**
  - Privileged server can replace failed components

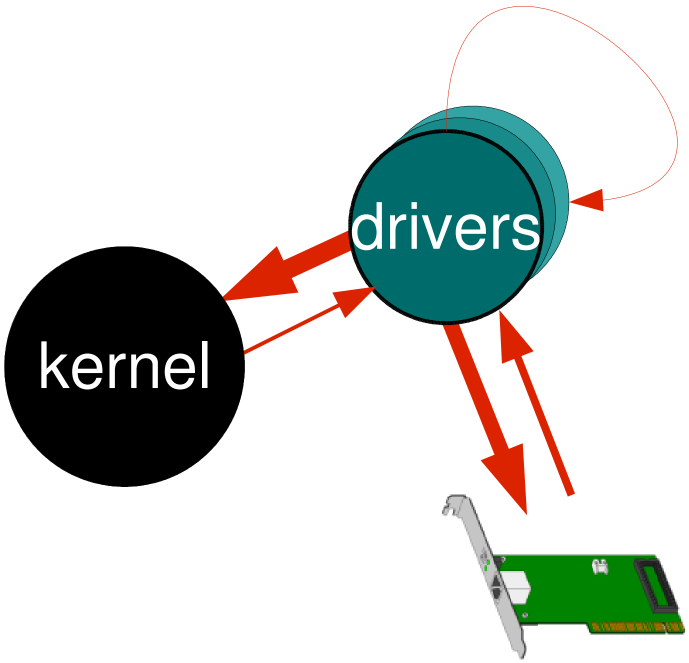
Jorrit N. Herder <jnherder@cs.vu.nl>

DRIVER–KERNEL DEPENDENCIES

- **Finding dependencies**
  - Compile driver code in isolation to find missing symbols
  - In addition, all drivers attempt to perform I/O

DRIVER–KERNEL DEPENDENCIES

- **Finding dependencies**
  - Compile driver code in isolation to find missing symbols
  - In addition, all drivers attempt to perform I/O
- **Who depends on what?**

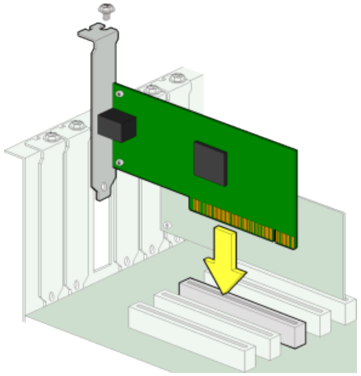


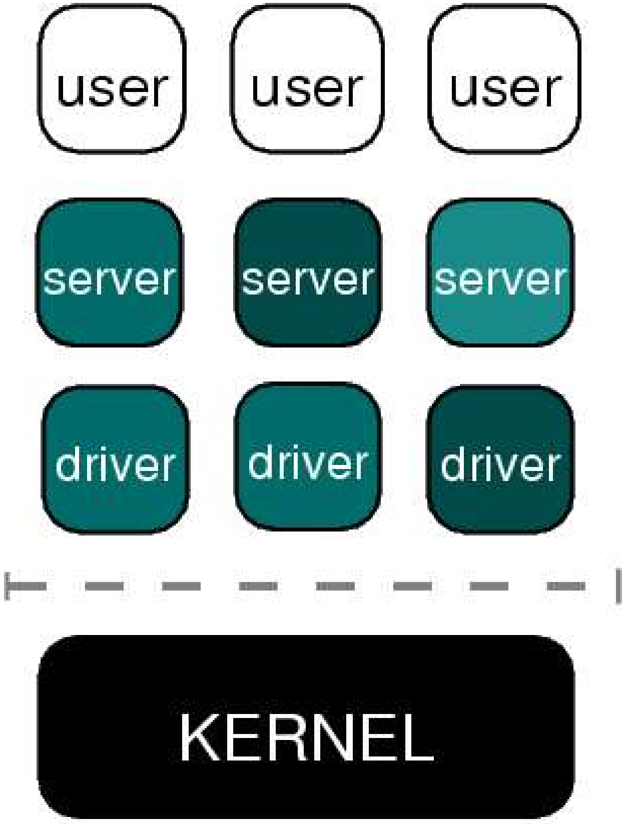
MOVING DRIVERS OUT OF THE KERNEL

- **Resolve dependencies one by one**
  - Add new system calls (SYS_DEVIO, etc.)
  - Disentangle interrupt handlers
  - Other improvements (new IPC, code cleanup, etc.)
- **Test modified driver in kernel space**
- **Finally, move to separate directory in user space**

ARCHITECTURE OF MINIX 3

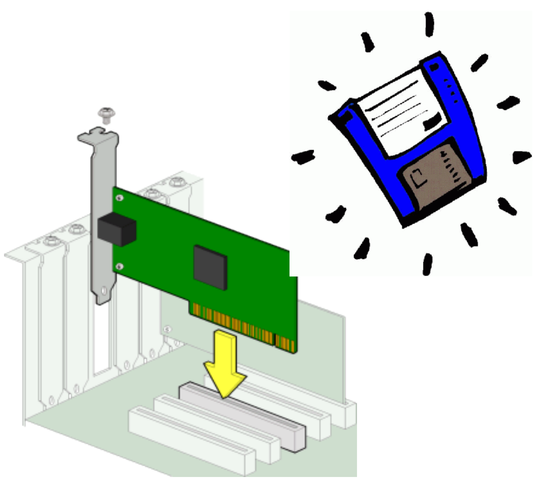
- **Device drivers are fully isolated in user space**

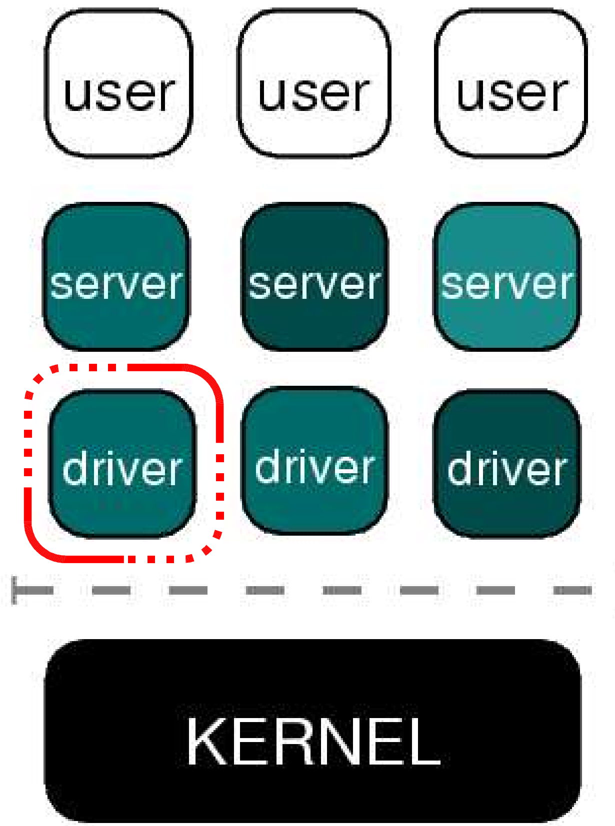




ARCHITECTURE OF MINIX 3

- Device drivers are fully isolated in user space
- Local failures cannot spread



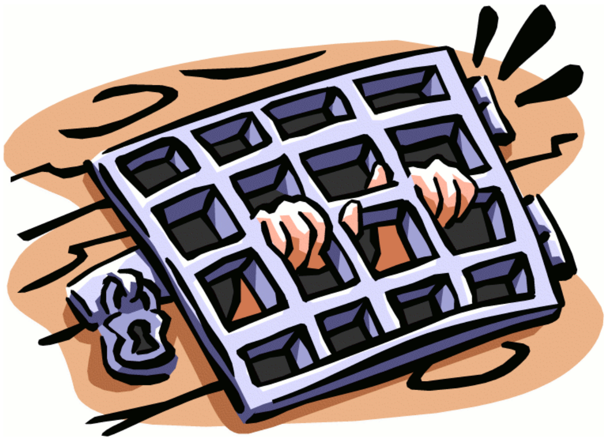


Jorrit N. Herder <jnherder@cs.vu.nl>

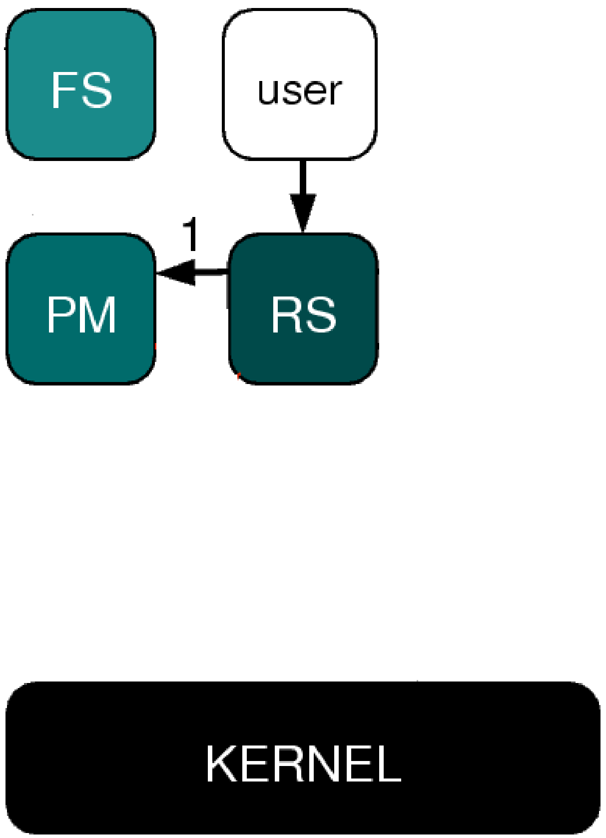
DEPENDABILITY

FAULT ISOLATION

- **All servers and drivers can fail independently**
- **Limit consequences of faults to enable recovery**
  - Servers and drivers fully compartmentalized in user space
  - Private address spaces protected by kernel and MMUs
  - Privileges of each process reduced according to POLA

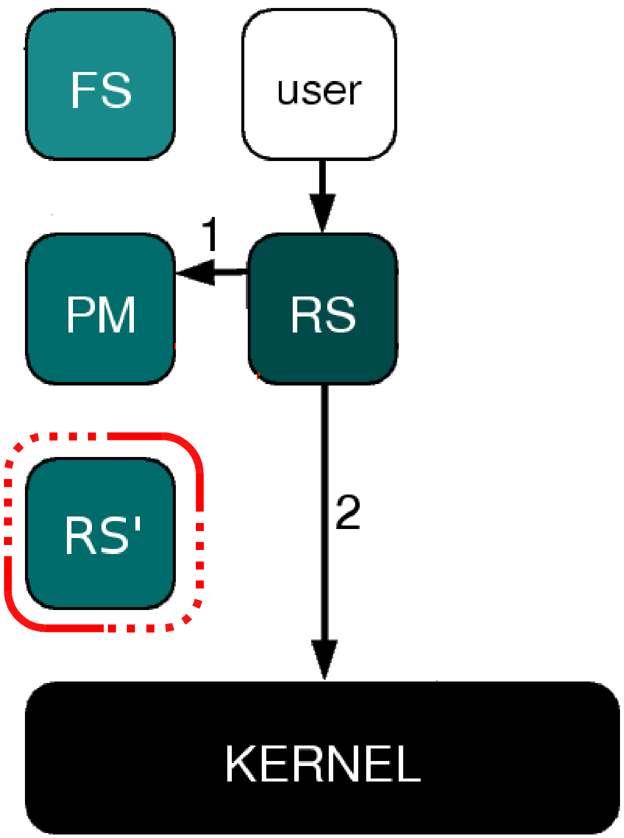


DEVICE DRIVER MANAGEMENT



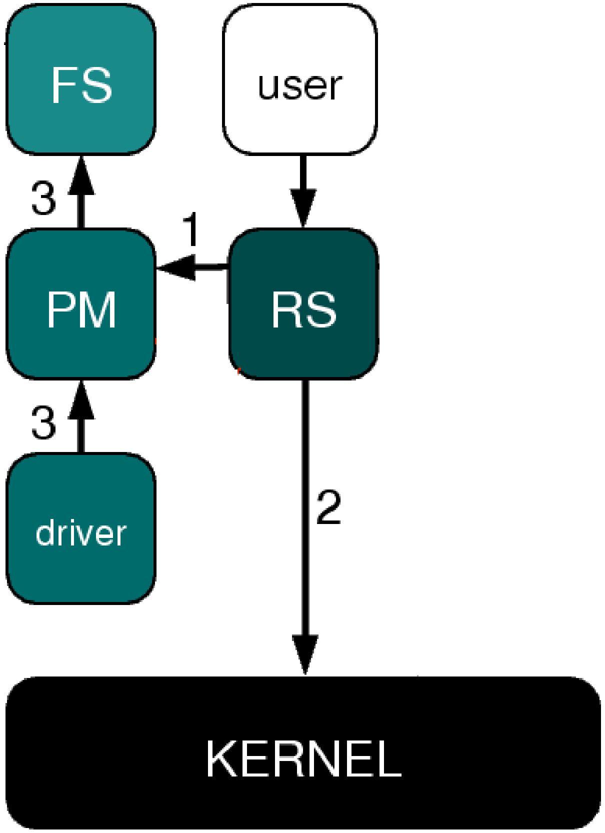
- **Starting a new driver**
  (1) Fork new process

DEVICE DRIVER MANAGEMENT



- **Starting a new driver**
  - (1) Fork new process
  - (2) Assign privileges

DEVICE DRIVER MANAGEMENT



- **Starting a new driver**

  (1) Fork new process

  (2) Assign privileges

  (3) Execute binary

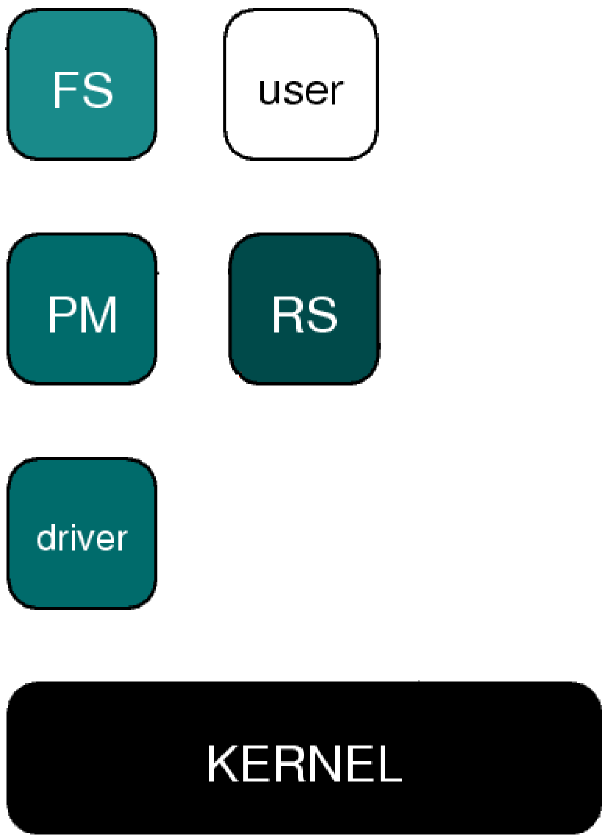
FAULT RESILIENCE

- **Our design tries to automatically *repair* defects**

  (1) Identify malfunctioning component

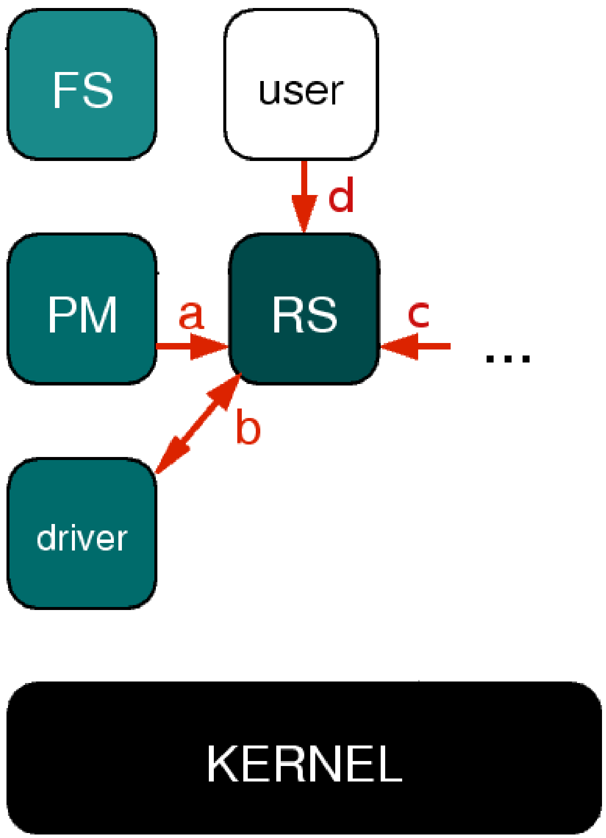
  (2) Execute associated recovery script

DETECTING DRIVER FAILURES



- **Human user observes**
  - System crash
  - Unresponsiveness
  - Weird behavior

DETECTING DRIVER FAILURES



- **OS monitors drivers**
  - (a) Exit notification
  - (b) Heartbeat message
  - (c) Component complains
  - (d) User requests update

RECOVERY PROCESS

- **Run recovery script**
  - Shell script that governs recovery steps taken
  - Full flexibility: write to log, send e-mail, restart component
- **Restart dead drivers**
  - Assumes restart enables recovery
- **Reintegrating the component**
  - Restarted component can retrieve lost state from data store
  - Dependent components are informed through data store

PERFORMANCE

Jorrit N. Herder <jnherder@cs.vu.nl>

PERFORMANCE OF MINIX 3

- **Overhead of user-mode drivers (compared to MINIX 2)**
  - Run times for typical applications: 6% overhead
  - File system and disk I/O performance: 9% overhead
  - Disk throughput (with fast disk and DMA) up to 70 MB/s
  - Networking performance: Fast Ethernet at full speed
    - Initial experiments show gigabit ethernet is possible
- **System feels fast and responsive**
  - Time from multiboot monitor to login is under 5 sec.
  - The system can do a full build of itself in under 10 sec.

SOURCE CODE STATISTICS

- **Kernel (including kernel tasks): < 4000 LOC**
- **Most important servers and drivers: ~2500 LOC**
- **Minimal POSIX-conformant system: ~20,000 LOC**
  - Critical source code reduced by >2 orders of magnitude
  - Sources are small enough to read and understand

DEPENDABILITY EVALUATION

- **Fault-injection experiments are work in progress**
- **Measurements of the recovery overhead:**

DISCUSSION

USER VIEW OF MINIX 3

- **Using MINIX 3 is like using a normal multiuser UNIX system**
  - However, not as mature as FreeBSD or Linux
  - Only 18 months of development with small core of people
    - Nevertheless, over 400 UNIX applications available
    - In-house TCP/IP stack with BSD sockets
    - X Window System was ported
    - VFS infrastructure was also added
    - VM support is next big hurdle

GENERAL APPLICABILITY

- **Users demand highly dependable systems**
  - Trade-off between “X” / dependability is changing
    - “X” = performance, costs, etc.
- **We offer a *useful* alternative to commodity systems**
- **Our techniques can be applied to other systems**
  - Trend towards user-mode drivers on other systems
  - Guard drivers similarly to what we have done

CONCLUSIONS

- **We have constructed a highly dependable OS**
  - Number of fatal (kernel) bugs is reduced
  - Isolation in user space limits bug damage
  - Recovery from common failures is possible
- **Our approach is practical for real-world adoption**
  - Overhead negligible compared to hardware improvements
  - Reduction of critical code base improves manageability
  - Fault injection experiments prove viability of approach

MORE INFORMATION

- Jorrit N. Herder, Herbert Bos, Ben Gras, Philip Homburg, Andrew S. Tanenbaum,

  Reorganizing UNIX for Reliability,

  Proc. 11th Asia-Pacific Computer Systems Architecture Conference, Shanghai, China, Sep. 2006.

- Jorrit N. Herder, Herbert Bos, Ben Gras, Philip Homburg, Andrew S. Tanenbaum,

  Construction of a Highly Dependable Operating System,

  Proc. 6th European Dependable Computing Conference, Coimbra, Portugal, Oct. 2006 .

TIME FOR QUESTIONS

- **Try it yourself!**
  - MINIX 3 Live CD-ROM
  - Current version: see website
- **More information**
  - Web: www.minix3.org
  - News: comp.os.minix
  - E-mail: jnherder@cs.vu.nl
- **The MINIX 3 team:**
  - Jorrit Herder
  - Mischa Geldermans
  - Ben Gras
  - Philip Homburg
  - Herbert Bos
  - Andy Tanenbaum

ANSWERS