AN OVERVIEW OF MINIX 3

NICTA seminar, 20 Oct. 2008

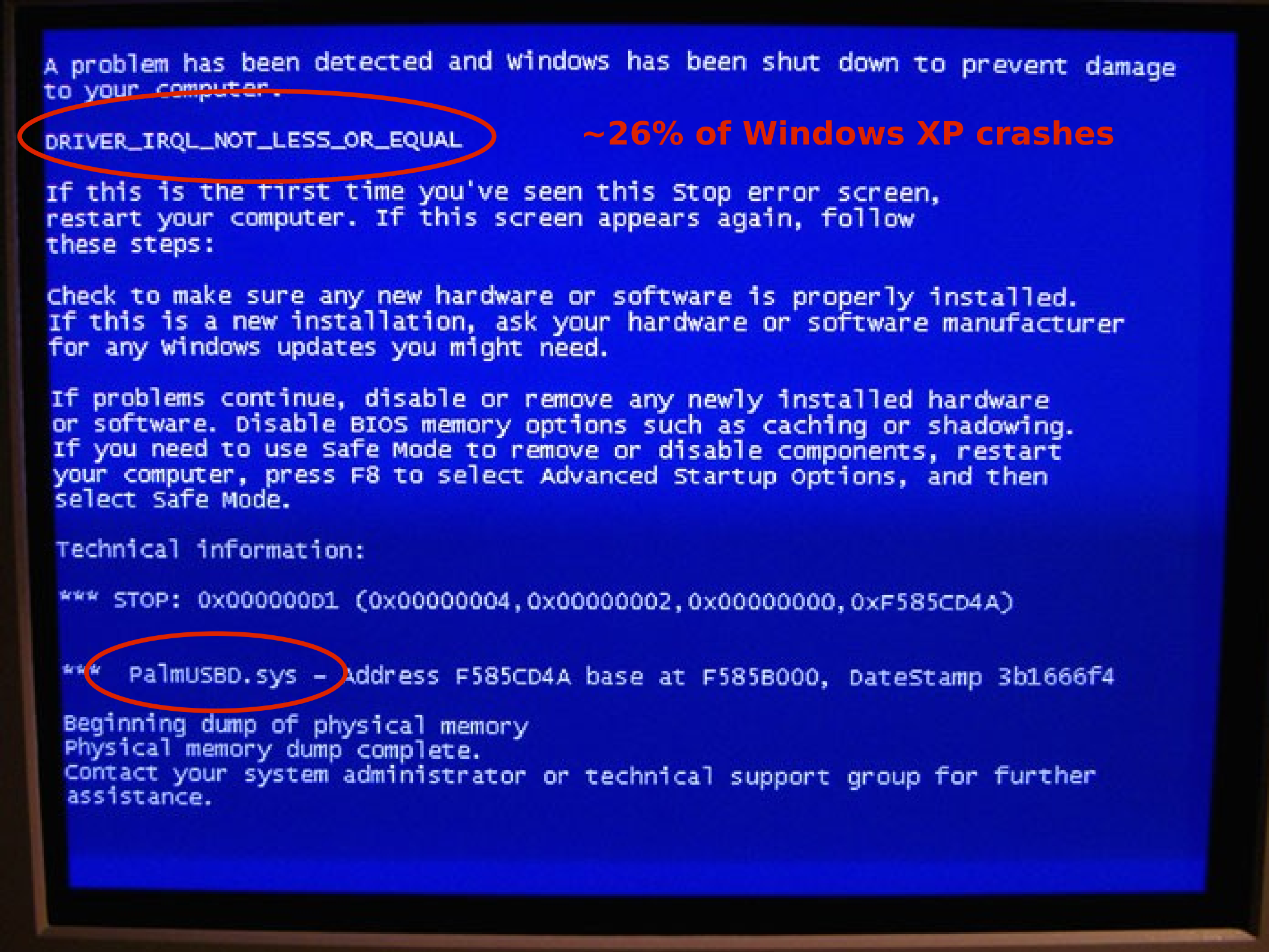
Sydney, Australia

Jorrit N. Herder

Vrije Universiteit Amsterdam

“There are no significant bugs in our released software that any significant number of users want fixed.”

-- Bill Gates, 1995


A problem has been detected and Windows has been shut down to prevent damage
to your computer.
DRIVER_IRQL_NOT_LESS_OR_EQUAL
~26% of Windows XP crashes
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.
Technical information:
*** STOP: 0x000000D1 (0x00000004,0x00000002,0x00000000,0xF585CD4A)
*** PalmUSBD.sys - Address F585CD4A base at F585B000, DateStamp 3b1666f4
Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.

Talk outline

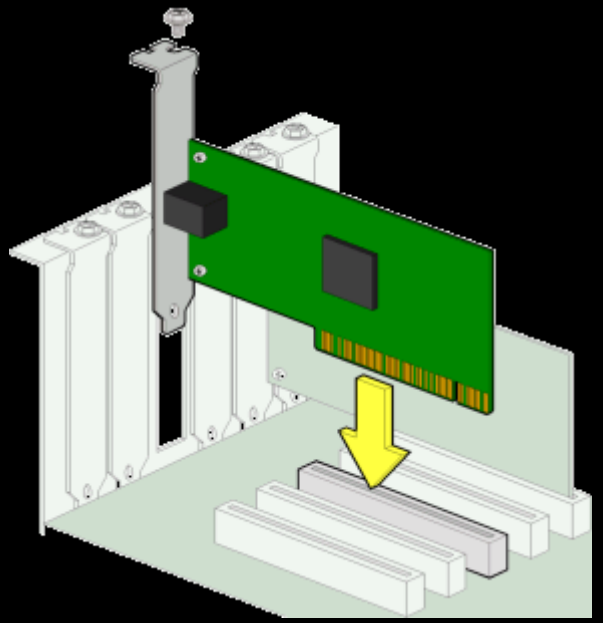
- Background and motivation
- MINIX 3 isolation architecture
- MINIX 3 self-repairing properties
- Experimental evaluation
- Discussion and conclusions

Talk outline

- Background and motivation
- MINIX 3 isolation architecture
- MINIX 3 self-repairing properties
- Experimental evaluation
- Discussion and conclusions

Even if OS were correct ...

- Plug-ins extend OS base functionality
  - provided by untrusted third parties
  - comprise up to 70% of entire OS
  - 3-7x more bugs than other OS code
- Still, extensions run in kernel
  - all powers of the system
  - no proper fault isolation

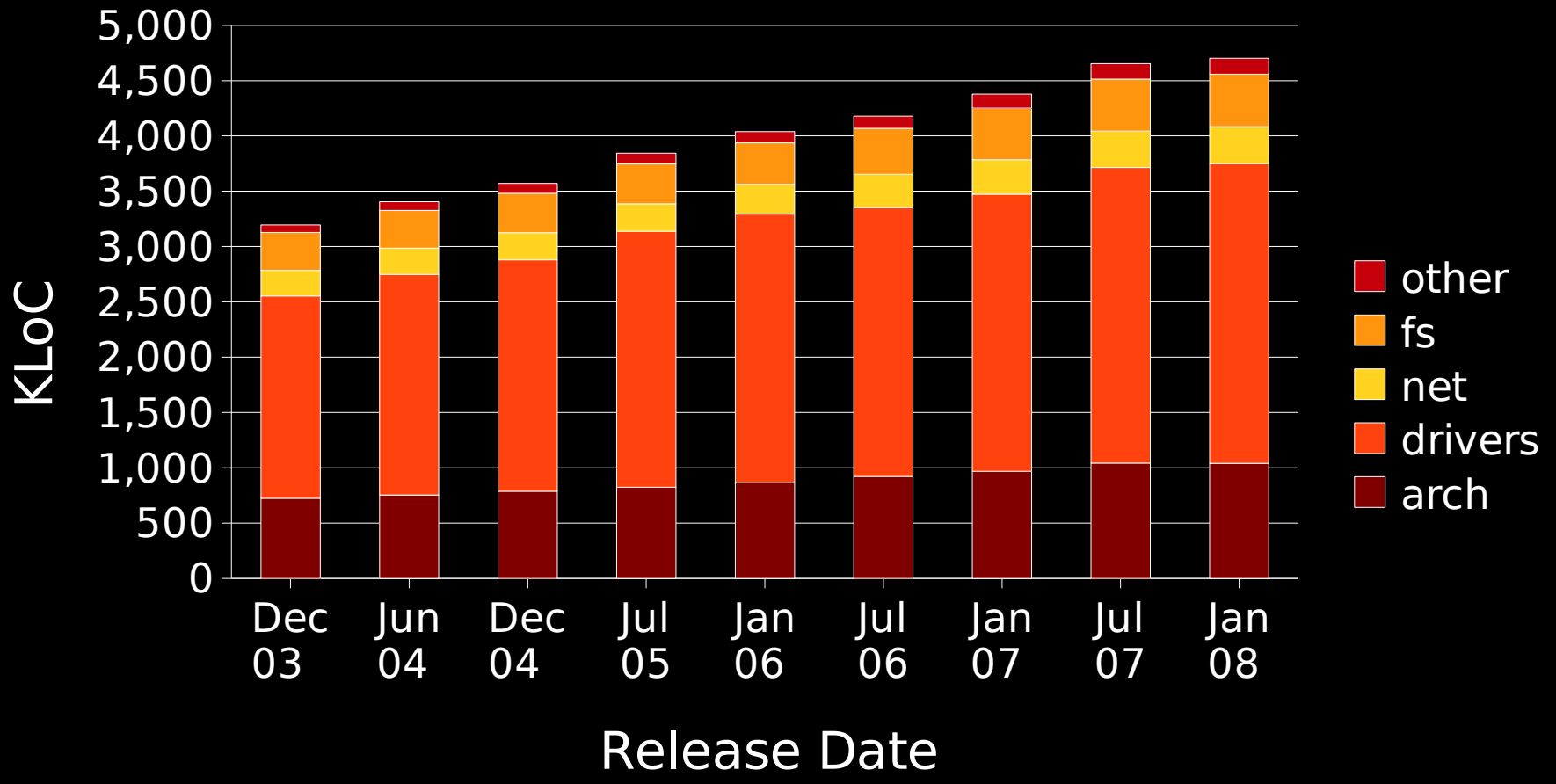


Bug fixing is infeasible

- Software is buggy by nature
  - survey across languages found 1-6 bugs/KLoc
  - e.g. FreeBSD: 3.35 post-release bugs/KLoC
- Continuously changing configurations
  - e.g. 88 new Windows drivers/day in 2004
- Code maintainability very hard
  - changing kernel interfaces
  - unwieldy growth of kernel code base

Linux 2.6 kernel analysis

- Sustained growth of ~5% every 6 months



Consequences

- Downtime mainly due to faulty software
  - over 50,000 kernel bugs in Linux/Windows
    - if kernel size estimated at ~5 MLoC
    - and fault density put at 10 bugs/KLoC
  - any kernel bug is potentially fatal

- Windows crash dump analysis confirms:
  - extensions cause 65-83% of all crashes

It's not *just* a nuisance

- Unacceptable for most (ordinary) users
  - grandma cannot handle computer crashes
- Big problem for large server farms
  - monthly reboot means many daily failures
    - even with 99% uptime a big problem
- Critical applications require reliability
  - ATMs, cars, power plants, etc.

MINIX 3 rationale

- Rethink OS design to improve reliability
  - no focus on performance as in L4
- But keep well-known UNIX “look and feel”
  - MINIX 3 is POSIX compliant UNIX clone
  - runs about 500 standard UNIX applications
    - standard shell, file, and text utilities
    - virtual file system infrastructure
    - TCP/IP stack with BSD sockets
    - X Window System

Talk outline

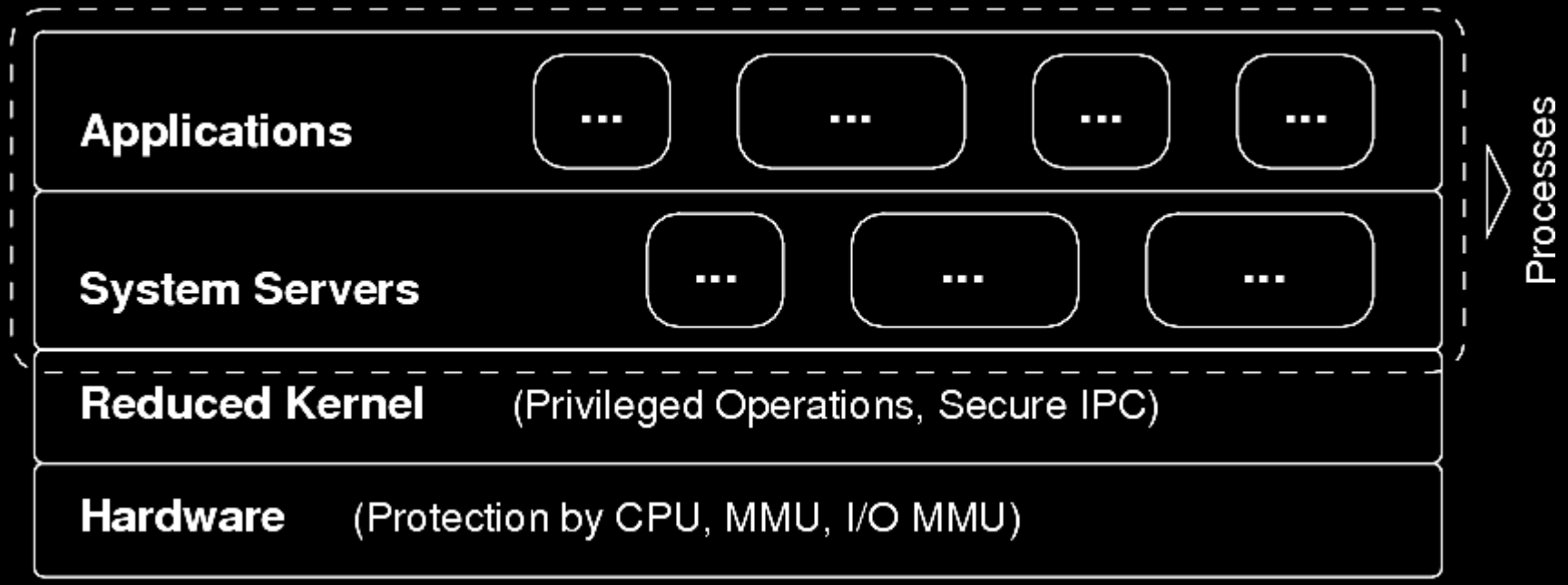
- Background and motivation
- MINIX 3 isolation architecture
- MINIX 3 self-repairing properties
- Experimental evaluation
- Discussion and conclusions

Isolation architecture

- Goal is to enforce least authority
  - only grant access required to do job
- This is realized using three techniques
  1) Structural constraints
     - run drivers as unprivileged processes
  2) Per-driver isolation policies
     - grant access to only resources needed
  3) Run-time memory granting
     - enable fine-grained data sharing

Structural constraints

- Multiserver design for all OS services
  - Process manager, virtual file system
  - Extension manager, file servers, drivers, etc.



Development benefits

- User-space components easily managed
  - Short development cycle
  - Normal programming model
  - No down time for crash and reboot
  - Easy debugging
  - Good flexibility

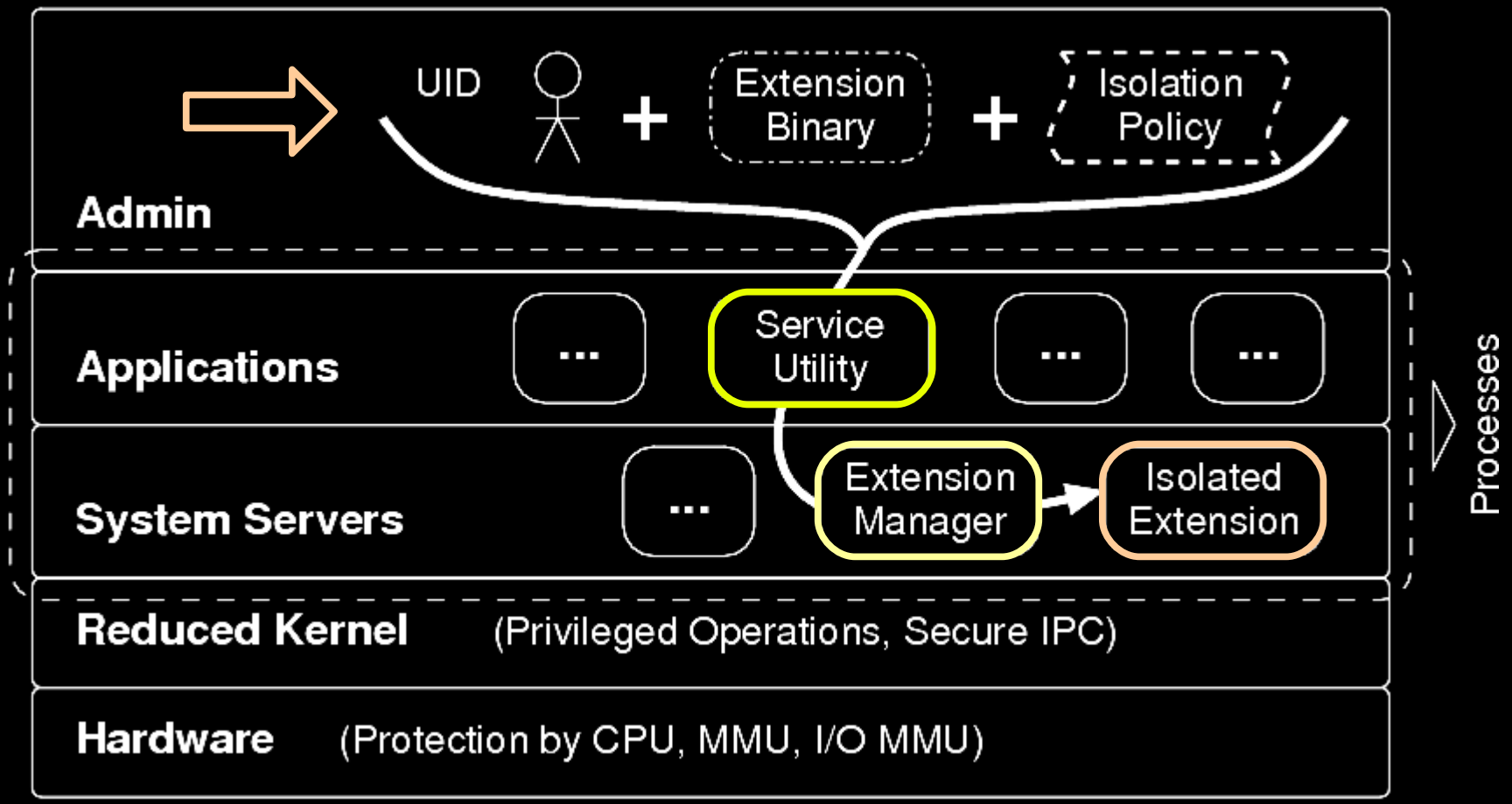
User-space drivers

- Only microkernel has full CPU privileges
  - manageable due to small size < 5,000 LoC
- Extensions run in user-space
  - unprivileged CPU mode
    - cannot change page tables, halt CPU, etc.
  - strict address-space separation
    - private, virtual address space

Kernel support needed

- Kernel mediates privileged operations
  - e.g., DEVIO kernel call mediates device I/O
    - based on resources granted in isolation policy
  - e.g., SAFECOPY call mediates memory copies
    - based on fine-grained memory grants
- Interrupt handling done in user-space
  - minimal, generic interrupt handler in kernel
    - structurally prevents 26% of Windows XP crashes

Per-driver isolation policies



Loading drivers in MINIX 3

- User sends request to extension manager
  - only privileged users may start extensions
- Actual loading and policy setting

  1) extension manager forks new process
     - IPC endpoint used to identify extension
  2) all OS servers are informed about policy
  3) finally process can execute binary

How to define policies?

- MINIX 3 policies are simple text files
  - each key grants one or more powers
    - e.g., 'ipc' denotes allowed IPC destinations
- Suitable for dynamic resource discovery
  - device resources not known in advance
    - PCI bus driver looks up device's resources

Example driver policy

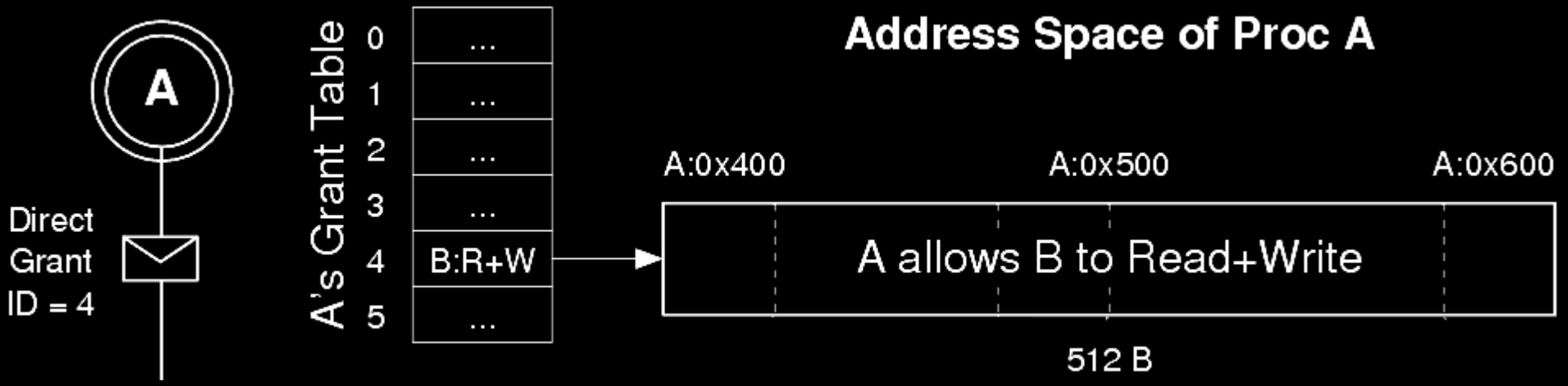
```
driver rtl8029 {                    # isolation policy
    pci device      10ec/8029;      # PCI RTL8029
    ipc             pci             # PCI bus driver
                    kernel;         # Kernel task
    ipc kernel      SAFECOPY        # Memory copying
                    DEVIO           # Device I/O
                    ...;
};
```

- Note: MINIX 3 provides only mechanisms

Run-time memory granting

- Address-space separation too strict
  - drivers typically need to exchange data
- Capability-like scheme: 'memory grants'
  - defines byte-granularity memory area
  - lists access rights for a specific process
  - grant validated by SAFECOPY kernel call
- Allow DMA into driver's address space
  - integration with grants is work in progress

Using memory grants



- Tell kernel about grant table location
- Add memory grant to allow access
- Pass index into grant table to grantee

Direct memory access (DMA)

- Prevent access to DMA controller
  - impractical for bus-mastering DMA
- Solution based on I/O MMU hardware
    - we used AMD's Device Exclusion Vector (DEV)
  - I/O MMU driver programs I/O MMU
  - allows DMA to only driver's address space

Talk outline

- Background and motivation
- MINIX 3 isolation architecture
- MINIX 3 self-repairing properties
- Experimental evaluation
- Discussion and conclusions

Self-repairing properties

- Isolation prevents fault propagation
  - cannot prevent buggy driver from failing
- Improve dependability through recovery
  - Failure is masked and OS can continue
  - Common in other areas:
    - Disks: Error correcting codes
    - Networking: Reliable protocols
    - Init process: respawns crashed daemons

Underlying idea

- Many faults tend to go away after restart
  - transient hardware faults
  - race condition due to timing issues
  - aging bugs causing failures over time
    - e.g., due to memory leaks

Detecting failures

- Human user observes
  - system crash / hang, weird behavior, etc.
- OS triggers for recovery
  - process exit due to internal panic
  - crashed by CPU or MMU exception
  - killed by user
  - heartbeat message missing
  - complaint by another component
  - dynamic update by user

Using recovery scripts

- Main benefit is full flexibility, e.g.:
  - log error messages
  - send e-mail to remote administrator
  - binary exponential backoff

```
if [ ! $reason -eq DYNAMIC_UPDATE ]; then
    sleep $((1 << ($repetition)))
fi
service restart $component
```

Restart procedure

- Driver manager starts new driver
- Pub-sub system broadcasts changes
- Dependent services recovery-aware
  - relatively small changes required
    - check for failures upon driver interactions
    - code to restart very similar to normal start
      - changes to INET limited to only 124 LoC
      - changes to VFS server limited to only 274 LoC
      - driver manager has 593 LoC relating to restarts

Effectiveness of recovery

- Recovery of device drivers
  - Network: transparent, no data loss
    - TCP protocol guarantees end-to-end integrity
  - Block: transparent, incidental data loss
  - Character: not transparent, data loss likely
- Recovery of system servers
  - yes, but result depends on state lost
    - restart of INET destroys all network sockets
    - still useful to improve uptime of web server

Talk outline

- Background and motivation
- MINIX 3 isolation architecture
- MINIX 3 self-repairing properties
- Experimental evaluation
- Discussion and conclusions

Performance considerations

- Modular design incurs some overhead
  - overhead can be limited to 5-10% range
    - long history of performance work in L4 context
- Our primary focus is dependability
  - rough estimate of current overhead 10-25%
  - MINIX 3 performs well for R&D purposes
    - e.g., full system build and restart under 15 sec
    - disk throughput up to 70 MB/s, limited by HDD
    - Fast Ethernet runs as full speed, limited by NIC

Raw disk throughput

- Sequential read for various I/O unit sizes
  - highlights worst-case overhead

Network driver recovery

- Repeated crashes simulated w/ SIGKILL
  - crash intervals ranging from 1 to 25 sec
- Transparent RTL8139 driver recovery
  - transparent recovery without data loss
  - mean recovery time is only 0.36 sec
    - 25% overhead with 1 crash every 1 sec
    - 8% overhead with 1 crash every 4 sec
    - 1% overhead with 1 crash every 25 sec
    - no overhead with no crashes

Fault-injection testing

- Goal: “Show that common OS errors in a properly isolated extension cannot propagate and damage the system.”

- Method: “Inject faults into an extension in order to induce a failure, and observe how the system is affected.”

Experimental setup

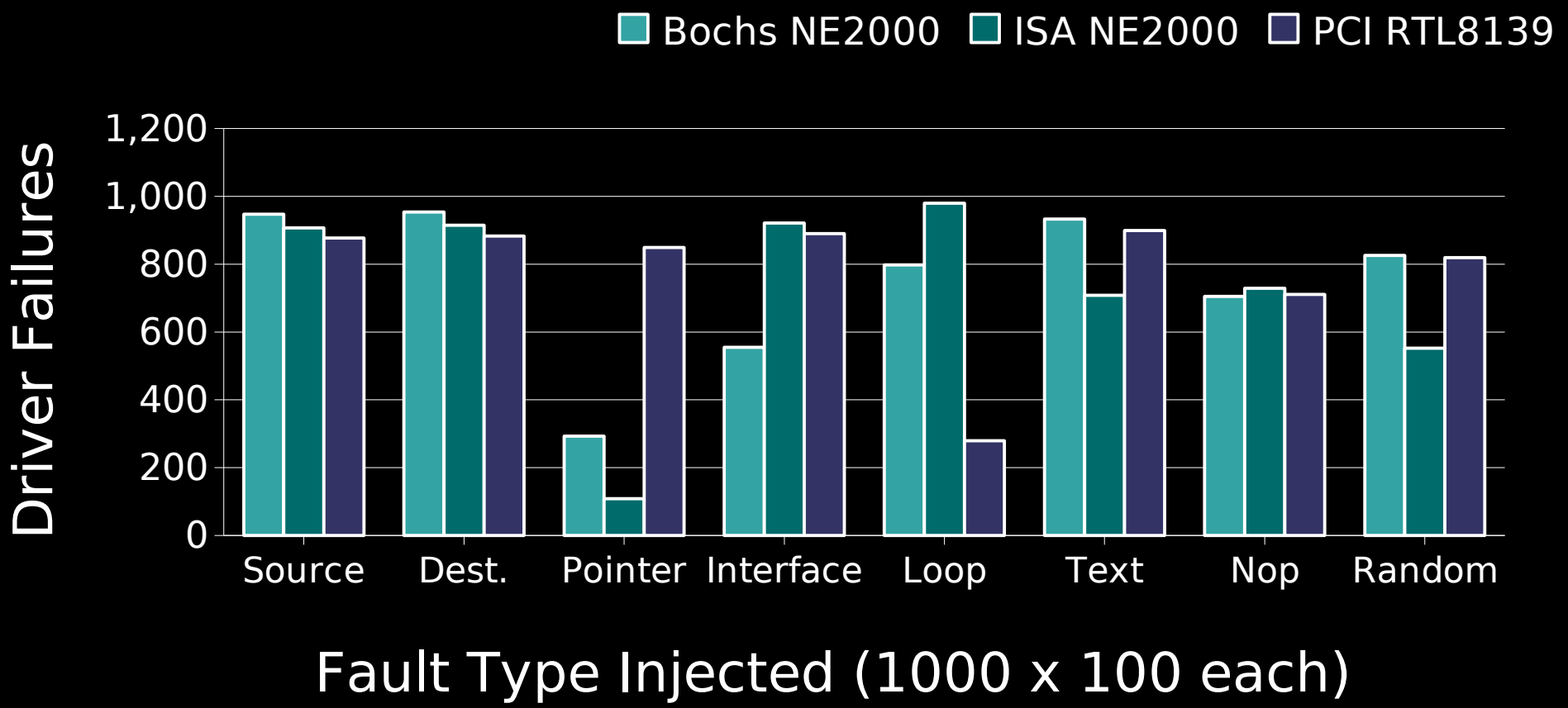
- Faults representative for common errors
  - e.g., bad pointers major crash cause (27%)
- Targeted drivers in networking subsystem
- Inject faults into text segment at run-time
  - workload exercised driver's functionality
- In total, we injected millions of faults
  - related work injected at most thousands
    - nasty bugs show up only after millions

Dependability results

- One experiment injected 2,400,000 faults
  - 3 configs * 8 types * 1000 trials * 100 faults
- This induced 18,038 detectable failures
    - CPU or MMU exceptions: 10,019
    - exits due to internal panic: 7,431
    - missing driver heartbeats: 588
- Transparent recovery in all 18,038 cases
    - “Failure Resilience for Device Drivers,” Proc. 37th DSN, pp. 41-50, June 2007

Distribution of failures

- Driver failed, but the OS never crashed



Engineering effort

- Statistics of executable source code

Component	# files	LoC	Comments
Fault injector	14	3,066	1,097
Extension manager	4	2,021	546
I/O MMU driver	1	329	10
PCI bus driver	3	2,798	339
VFS server	24	6,050	2,434
SATA driver	3	2,443	851
TCP/IP server	53	20,033	1,691
RTL8139 driver	1	2,398	345
NE2000 driver	3	2,769	424
Microkernel	54	4,753	2,600

Talk outline

- Background and motivation
- MINIX 3 isolation architecture
- MINIX 3 self-repairing properties
- Experimental evaluation
- Discussion and conclusions

Conclusions (1/3)

- Extensions threaten OS dependability
  - unwieldy driver growth not manageable
  - isolation of untrusted code is needed
- MINIX 3 employs a multiserver design
  - rethink OS internals to isolate extensions
  - but keep UNIX “look and feel” for users

Conclusions (2/3)

- MINIX 3 isolation architecture
  - structural restrictions
  - per-driver isolation policy
  - run-time memory granting
- MINIX 3 self-repairing properties
  - monitor driver failures at run-time
  - script-driven recovery procedure

Conclusion (3/3)

- Sacrifice performance for dependability
  - current overhead estimated at 10-25%
  - optimizations possible, as shown by L4
- Fault-injection testing proves viability
  - injected over 2,400,000 common OS faults
  - 18,038 driver failures that could be recovered
  - MINIX 3 operating system never crashed

Acknowledgements

- NICTA
  - Gernot Heiser
  - Emilia Sotirova
- The MINIX 3 team
  - Ben Gras
  - Arun Thomas
  - Philip Homburg
  - Herbert Bos
  - Andy Tanenbaum

Thank you! Questions?

- Try it yourself!
  - download MINIX 3

    - www.minix3.org
- More information:
  - web: www.minix3.org
  - news: comp.os.minix
  - e-mail: jnherder@cs.vu.nl