FAILURE RESILIENCE FOR DEVICE DRIVERS

37th Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks

Edinburgh, UK
June 25-28, 2007

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

“The unavoidable price of reliability is simplicity.”

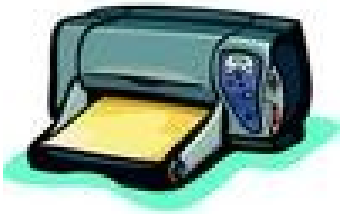
--- C.A.R. Hoare

WHAT IS THIS ABOUT?

- **Failure resilience improves dependability**
  - **Resilience means quick recovery from failures**
  - **Failure is masked and system can continue**
- **Common in other areas, for example:**
  - **RAID: overcome drive failures**
  - **ECC memories: correct bit errors**
  - **Disks, CD-ROMS, DVDs: prevent corruption**
  - **TCP: handles lost, misordered, garbled packets**
  - **Init process: respawns crashed daemons**

FAILURE RESILIENCE FOR DEVICE DRIVERS

- We want to extend this idea to OS internals
  - Tolerate certain failures in OS extensions
  - Focus is on restarting drivers after a crash
- Extensions are often provided by third parties
  - Typically comprise 70% of operating system code
  - Reported error rates 3-7x higher than other code
  - E.g., 85% of Windows XP crashes due to drivers
- Quick path to increased OS dependability!



TALK OUTLINE

- Introduction (continued)
- Recovery procedure
- Driver recovery schemes
- Experimental evaluation
- Conclusions

INTRODUCTION (CONTINUED)

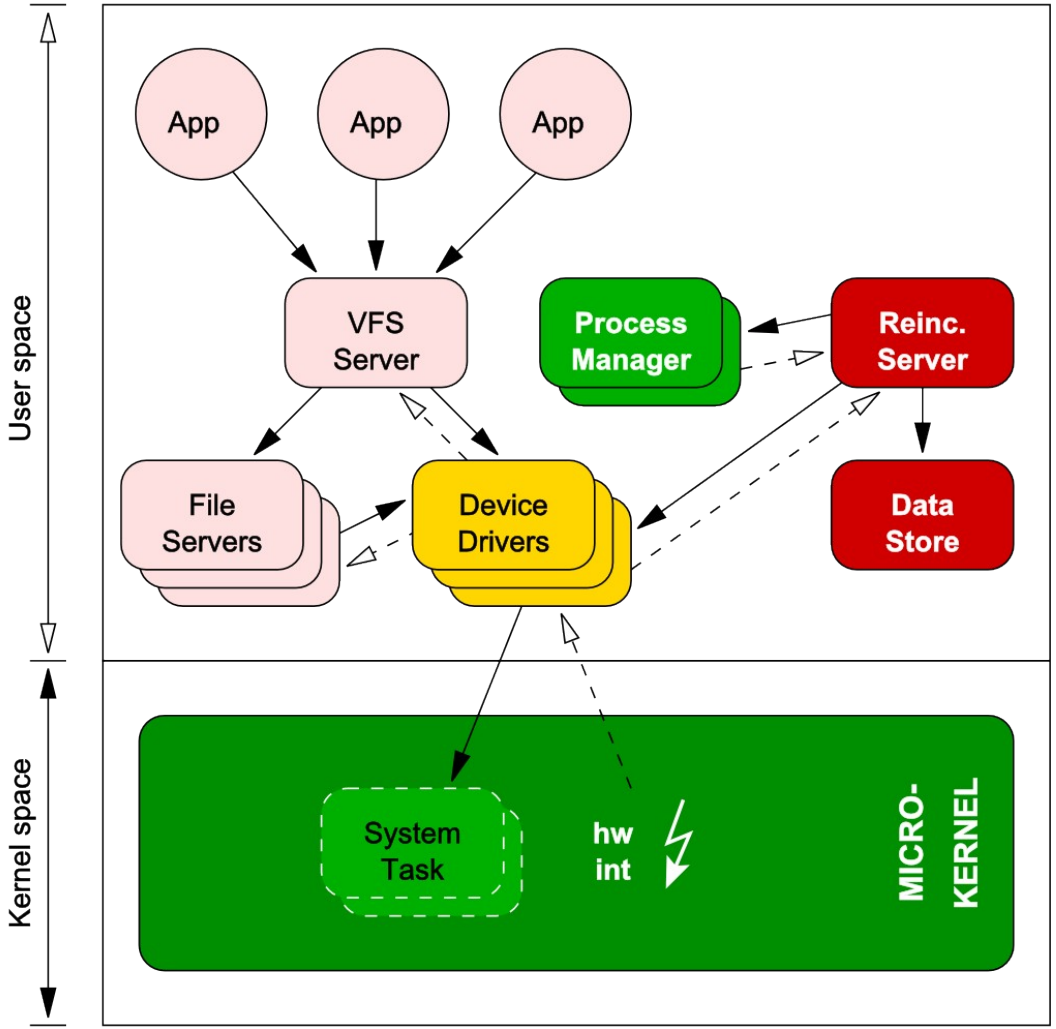
FAILURE MODEL

- Focus on driver failures; but not errors and faults
- Focus on intermittent and transient driver failures
  - Exceptions triggered by unexpected input
  - Panics due to internal inconsistencies
  - Race conditions caused by rare hardware timing
  - Aging bugs that caused failure over time
- Hard to track down, but often cured by restart
  - Restart only failed component after detection

ISOLATION ARCHITECTURE

- **Crucial prerequisite for failure resilience**
  - **Prevent problems from spreading throughout**
  - **All servers and drivers can fail independently**
- **Drivers compartmentalized in user space**
  - **Separate processes with private address space**
  - **Privileges of drivers reduced according to POLA**
    - Unprivileged user and group ID
    - IPC primitives and IPC targets
    - Kernel calls
    - I/O ports and IRQ lines allowed

OVERVIEW OF SYSTEM ARCHITECTURE



- **Reincarnation Server**
  - Manage drivers
  - Monitor system
  - Repair defects
- **Data Store**
  - Publish configuration
  - Backup state

RECOVERY PROCEDURE

DEFECT DETECTION

- System's well-being is constantly monitored
- Inputs that trigger recovery procedure:

  (1) Process exit or panic

  (2) Crashed by CPU or MMU exception

  (3) Killed by user

  (4) Heartbeat message missing

  (5) Complaint by another component

  (6) Dynamic update by user

POLICY-DRIVEN RECOVERY

- Drivers associated with policy script
- General recovery steps taken after a failure
  (1) Malfunctioning component is identified
  (2) Associated policy script is run with context
  (3) Component may be replaced with a fresh copy
- Precise action based on script's context
  - Component that failed
  - Kind of failure
  - Failure count
  - Script parameters

USING POLICY 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
```

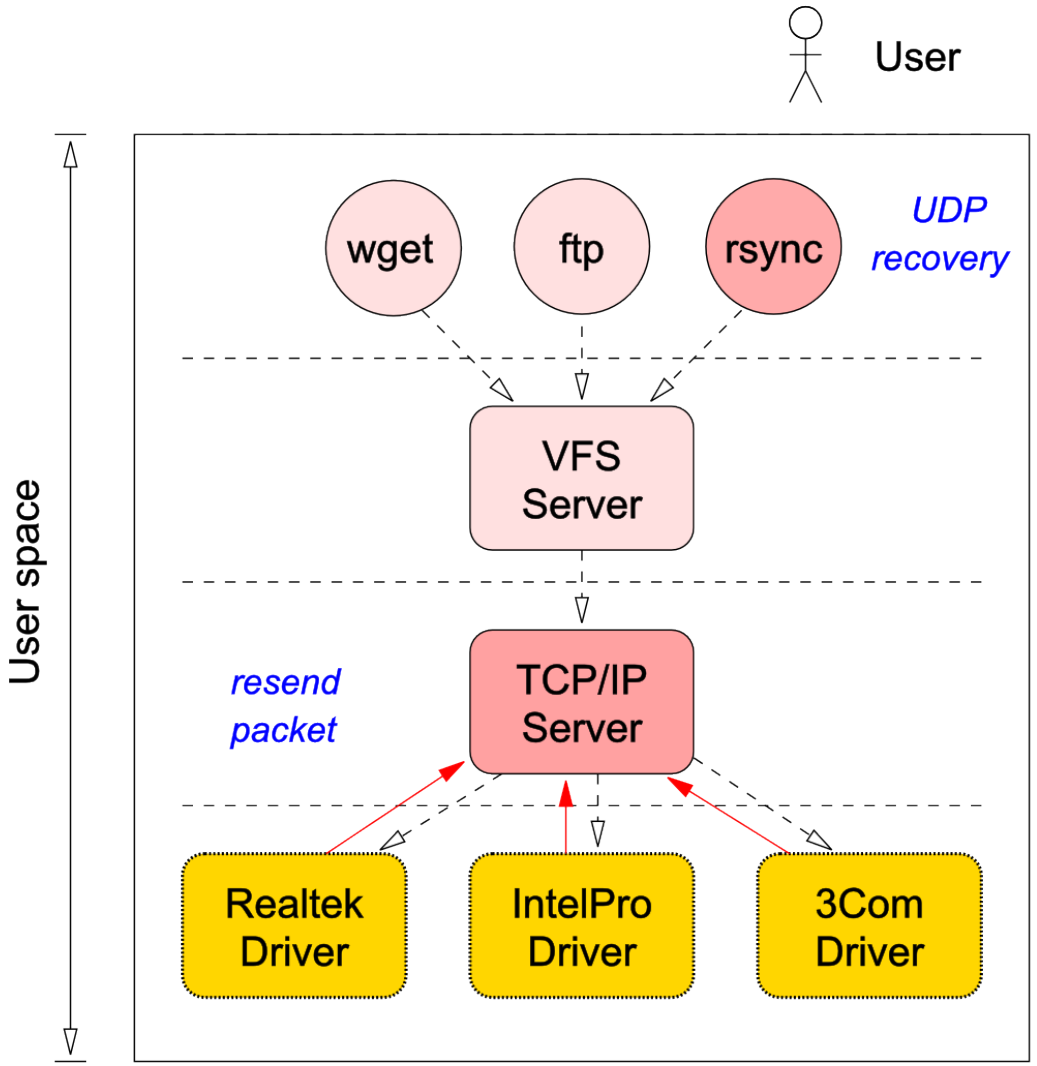
DRIVER RECOVERY SCHEMES

DRIVER RECOVERY SCHEMES

- **Transparent driver recovery is often possible**
  - **Network driver recovery :: yes (*)**
  - **Disk driver recovery :: yes**
  - **Character driver recovery :: maybe (*)**
- **Recovery of failed servers**
  - **Sometimes possible, depending on lost state**

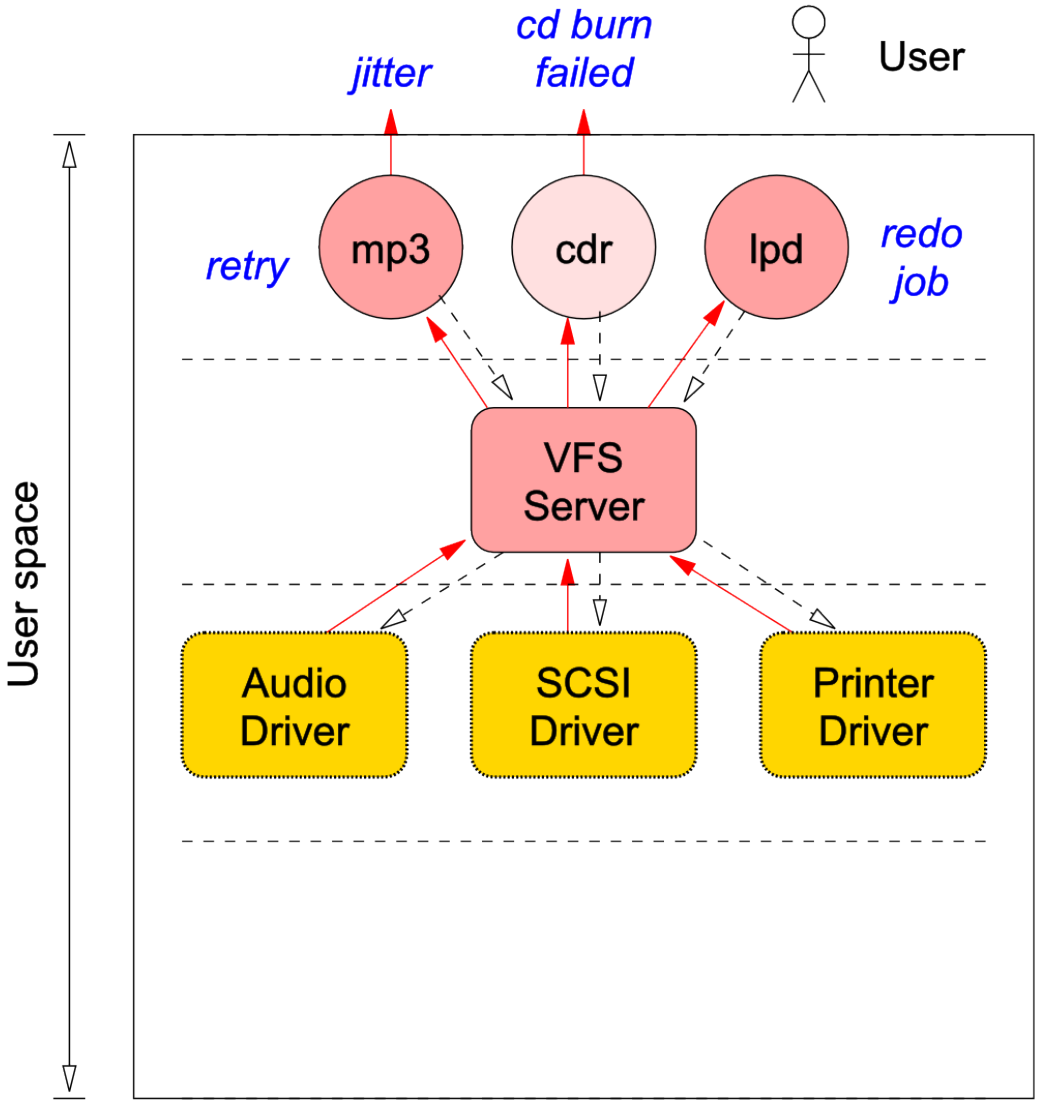
(*) = detailed on following slides

NETWORK DRIVER RECOVERY



- **Transparent recovery**
  - **TCP protocol**
    - **Mark requests pending**
    - **TCP handles data loss**
  - **UDP protocol**
    - **Depends on UDP client**

CHARACTER DRIVER RECOVERY



- **Not transparent**
  - **Data stream interrupted**
    - **Recovery at application**
    - **Error reported to user**
- **Change applications**
  - **Retrying I/O possible**

EXPERIMENTAL EVALUTION

PERFORMANCE OVERHEAD

- Simulated repeated crashes using SIGKILL
  - Crash interval intervals ranging from 1 to 25 sec
- Transparent RTL8139 driver recovery
  - Transparent recovery was successful in all cases
  - Mean recovery time is 0.36 sec due to TCP timeouts
    - 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

- SWIFI to simulate real-life failures
  - Mutate binary code of running driver
  - 7 fault types representative for common OS errors
- Transparent DP8390 driver recovery in Bochs
  - Over 12,500 random faults led to 347 crashes
    - 226 exits due to internal panic
    - 109 kills due to CPU or MMU exception
    - 12 restarts due to missing heartbeat
  - Restart successful in 100% of the induced failures
- Real hardware showed 99% success rate thus far
  - Hardware limitations required low-level BIOS reset

REENGINEERING COSTS

- **Changes are both limited and local**
  - **Integrated approach required for optimal results**
  - **Most driver changed only few lines of shared code**
  - **Changes to limited to few key components, e.g.:**
    - **Reincarnation server: 2002 LoC (+ 593 LoC)**
    - **Data store: 384 LoC (+ 59 LoC)**
    - **VFS server: 5464 LoC (+ 274 LoC)**
    - **Network server: 20019 LoC (+ 124 LoC)**

CONCLUSIONS

CONCLUSIONS

- We have built a failure-resilient OS
  - Aimed at transient and intermittent driver failures
  - Transparent recovery is often possible
  - Recovery scheme depends on type of driver
- We have provided a concrete evaluation
  - Fault-injection and crash simulation prove viability
  - Performance overhead of recovery is very limited
  - Limited engineering costs compared to driver code base
- We believe our approach is practical as well
  - Ideas can be applied to other systems, like Windows

ACKNOWLEDGEMENTS

- The MINIX 3 team
  - Ben Gras
  - Philip Homburg
  - Herbert Bos
  - Andy Tanenbaum

TIME FOR QUESTIONS

- **Availability**
  - On the spot: MINIX 3.1.3 CD-ROM
  - Web: www.minix3.org
  - News: comp.os.minix
  - E-mail: jnherder@cs.vu.nl