Recovering Device Drivers

Mike Swift, Muthu Annamalai, Brian Bershad, Hank Levy

University of Washington
Device Drivers Cause Crashes

- Device drivers are the most common cause of system crashes
  - 85% of Windows XP crashes caused by drivers
  - Linux drivers 7x buggier than other kernel code

- System reliability will not improve until we fix the driver problem
Driver Crashes

Kernel
Driver
Application

Application

Application

Driver Crashes
SOSP 2003: Isolating Drivers

Restarting failed drivers prevents system crashes by reinitializing driver & kernel data structures
SOSP 2003: Isolating Drivers

Restarting does not prevent application crashes

- Loses application state in driver
- Exposes application to errors during restart
Preventing Application Crashes

1. Rewrite driver to recover itself
Preventing Application Crashes

1. Rewrite driver to recover itself
2. Rewrite applications to handle driver failures
Preventing Application Crashes

1. Rewrite driver to recover itself
2. Rewrite applications to handle driver failures
3. Conceal driver failures with a generic recovery service
Generalizations About Drivers

1. Rebooting fixes failures
   - Focus on transient errors
Generalizations About Drivers

1. Rebooting fixes failures
   - Focus on transient errors
2. They can be made to fail cleanly
   - Recover by restarting driver
Generalizations About Drivers

1. Rebooting fixes failures
   - Focus on transient errors
2. They can be made to fail cleanly
   - Recover by restarting driver
3. Small # of common interfaces
   - Leverage well-known behavior without knowledge of implementation
Outline

- Introduction
- The Shadow Driver System
  - Overview
  - Components
- Evaluation
- Conclusion
Shadow Driver Overview

- Shadow drivers hide driver failures from applications and the OS
  - Generic service infrastructure
  - Leverages existing driver/kernel interface
  - One shadow driver handles recovery for an entire class of device drivers
Shadow Driver Overview

- Shadow drivers hide driver failures from applications and the OS
  - Generic service infrastructure
  - Leverages existing driver/kernel interface
  - One shadow driver handles recovery for an entire class of device drivers

- What shadow drivers do:
  - Prepare
  - Recover
  - Conceal
Today’s Systems

Kernel

Device Driver

write(...)  register(...)
Shadowing a Working Driver

Kernel

Device Driver

Tap

Shadow Driver

write(...)
Shadowing a Working Driver

Kernel

Tap

Device Driver

Shadow Driver

done(...)
Spoofing a Failed Driver

Kernel → Tap → Shadow Driver

write(…)

Device Driver

write(…)
Recovering a Failed Driver

Kernel

Tap

Device Driver

Shadow Driver

register(...)
What Shadow Drivers Do

- **Prepare:**
  - Monitor kernel-driver communication

- **Recover:**
  - Restart driver after failure

- **Conceal:**
  - Act as driver during recovery
Preparing for Recovery

- Monitor kernel-driver communication to capture relevant state
  - Configuration operations
  - Active connections
  - Outstanding requests
1. Reset driver
2. Repeat driver initialization calls
3. Transfer in state
   - Reopen active connections
   - Replay configuration requests from log
   - Resubmit active requests

Recovering Driver
Recovering Driver

1. Reset driver
2. Repeat driver initialization calls
3. Transfer in state
   - Reopen active connections
   - Replay configuration requests from log
   - Resubmit active requests

- Shadow responds to driver’s kernel requests
  - Hide restart from kernel and driver
  - Supply driver with existing resources
Concealing Failure

- Shadow acts as driver
  - Applications and OS unaware that driver failed
  - No device control
- General Strategies:
  1. Answer request from log
  2. Act busy
  3. Block caller
  4. Queue request
  5. Drop request
Implementation

- Implemented in Linux 2.4.18 kernel
- Uses Nooks driver fault isolation system
- Supports three driver classes:
  - Sound card
  - Network interface card
  - IDE storage
Outline

- Introduction
- Shadow Driver System
- Evaluation
  - Can shadow drivers conceal failure?
  - At what cost?
    - Performance
    - Complexity
- Conclusion
# Drivers Tested

<table>
<thead>
<tr>
<th>Class</th>
<th>Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sound</td>
<td>SoundBlaster Audigy, Soundblaster Live!, Intel 810 Audio, Ensoniq 1371, Crystal Sound 4232</td>
</tr>
<tr>
<td>Network</td>
<td>Intel Pro/1000 Gigabit Ethernet, Intel Pro/100 10/100, 3Com 3c59x 10/100, AMD PCnet32, SMC Etherpower 100</td>
</tr>
<tr>
<td>IDE Storage</td>
<td>ide-disk, ide-cd</td>
</tr>
</tbody>
</table>
Evaluation

- **Testing Methodology**
  - Add bugs to driver
    - Port real bugs
    - Inject synthetic bugs
  - Run application using driver

- **Platforms:**
  - *Native:* standard 2.4.18 kernel
  - *Shadow:* fault isolation + shadow drivers
## Possible Outcomes

<table>
<thead>
<tr>
<th></th>
<th>Everything kept working</th>
<th>Application crashed</th>
<th>Total system crash</th>
</tr>
</thead>
<tbody>
<tr>
<td>![GreenTick]</td>
<td>![RedX]</td>
<td>![RedXX]</td>
<td></td>
</tr>
<tr>
<td>App.</td>
<td>Native</td>
<td>Shadow</td>
<td>SOSP</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>Mp3 Player</td>
<td>XX</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Audio Recorder</td>
<td>XX</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Speech Synth.</td>
<td>XX</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Game</td>
<td>XX</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Remote Copy</td>
<td>XX</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Remote Window</td>
<td>XX</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Packet Sniffer</td>
<td>XX</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Compiler</td>
<td>XX</td>
<td>✓</td>
<td>XX</td>
</tr>
<tr>
<td>Encoder</td>
<td>XX</td>
<td>✓</td>
<td>XX</td>
</tr>
<tr>
<td>Database</td>
<td>XX</td>
<td>✓</td>
<td>XX</td>
</tr>
</tbody>
</table>
Large-Scale Fault Injection

Percent of Failures

Recovered

MP3 Player
Audio Recorder
Remote Copy
Sniffer
Compiler
Database

Sound
Net
Storage
Large-Scale Fault Injection

- Automatic Detection
- Manual Detection

Percent of Failures

- Sound
- Net
- Storage

Components:
- Mp3 Player
- Audio Recorder
- Remote Copy
- Sniffer
- Compiler
- Database
Large-Scale Fault Injection

- Automatic Detection
- Manual Detection
- Failed Recovery

Percent of Failures

- Mp3 Player
- Audio Recorder
- Remote Copy
- Sniffer
- Compiler
- Database

Storage

Sound

Net
Relative Performance

Native vs Shadow for various tasks:
- Mp3 Player
- Audio Recorder
- Network Send
- Network Receive
- Compiler
- Database

Performance metrics include:
- Sound
- Storage

Relative Performance (%)
## Complexity

<table>
<thead>
<tr>
<th>Driver Class</th>
<th>Shadow Driver L.O.C.</th>
<th>1 Device Driver L.O.C.</th>
<th>All Drivers Count</th>
<th>All Drivers L.O.C.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sound</td>
<td>666</td>
<td>7,381</td>
<td>48</td>
<td>118,981</td>
</tr>
<tr>
<td>Network</td>
<td>198</td>
<td>13,577</td>
<td>190</td>
<td>264,500</td>
</tr>
<tr>
<td>Storage</td>
<td>321</td>
<td>5,358</td>
<td>8</td>
<td>29,000</td>
</tr>
</tbody>
</table>

- Shadow Drivers: 3300 lines
- Nooks Fault Isolation: 23,000 lines
- Linux Kernel: 2.7 million lines
Conclusion

- Shadow drivers protect applications from driver failures
  - Shadow drivers leverage existing driver interfaces for recovery
  - Shadow drivers prevented 98% of application failures in testing
- Shadow drivers have low cost
Want More Information?

mikesw@cs.washington.edu
nooks.cs.washington.edu

or

invite me for an interview