Improving the Reliability of Commodity Operating Systems

Mike Swift, Brian Bershad, Hank Levy
University of Washington
Outline

• Introduction
• Vision
• Design
• Evaluation
• Summary
The Problem

• Operating system crashes are a huge problem today
  – 5% of Windows systems crash every day
• **Device drivers** are the biggest cause of crashes
  – Drivers cause 85% of Windows XP crashes
  – Drivers are 7 times buggier than the kernel in Linux

• We built Nooks, a system that prevents drivers from crashing the OS
  – We can prevent 99% of faults in our tests that crash native Linux
Crashes Today

User Program

User Program

Kernel

Driver
Crashes Today

User Program  
Kernel  
Driver  
User Program
Crashes Today

- User Program
- Kernel
- Driver
- User Program
Outline

• Introduction
• Vision
• Design
• Evaluation
• Summary
Vision

User Program

Kernel

Driver

User Program
Reality

• **Windows XP**
  – 113 million copies sold in 2002
  – 40 million lines of code
  – $1 billion development cost
  – 35,000 drivers available

• **Linux:**
  – 18 million users
  – 30 million lines of code
  – Equivalent $1 billion development cost
Vision Requirements

1. Isolation
2. Recovery
3. Compatibility
   • No code changes
   • No new languages
   • No new OS
   • No new hardware
   • No new perspective
Outline

• Introduction
• Vision
• Design
• Evaluation
• Summary
Assumptions and Principles

• Assumptions:
  – Drivers are generally well behaved
  – Don’t need to prevent every crash to be useful

• Principles:
  – Design for fault resistance (not fault tolerance)
  – Design for mistakes (not abuse)
Goal

We want a practical, “best-effort” solution
• Prevents many crashes
• Good performance
• Works with today’s operating systems and drivers
Design of Nooks

• Standard Linux kernel and drivers
• Plus:
  – Isolation
  – Recovery
• Compatible with existing code
Existing Kernels

User Program

User Program

Kernel

Driver
Isolation - Memory

User Program

Kernel

User Program

Driver

Stack

Heap

Lightweight Kernel Protection Domains
Isolation - Control Transfer

- User Program
- User Program
- Kernel
- Driver
Isolation - Control Transfer

User Program

Kernel

User Program

eXtension Procedure Call

Driver

XPC

XPC
Isolation - Data Access
Isolation - Data Access

User Program

Kernel

Driver

Copy-in / Copy-out
Isolation - Interposition

User Program

Kernel

Driver

User Program
Isolation - Interposition

User Program

Kernel

Driver

XPC Wrappers
Design Summary

• Isolation
  – Lightweight Kernel Protection Domains
  – eXtension Procedure Call (XPC)
  – Copy-in/Copy-out
  – Wrappers
Recovery - Fault Detection

- Recovery
- User Program
- Kernel
- Driver
- Processor
Recovery - Fault Detection

User Program

Kernel

Recovery

Driver
Recovery - Fault Detection
Recovery

User Program

User Program

Kernel

Recovery

Stop / Unload
Design Summary

• Isolation
  – Lightweight Kernel Protection Domains
  – eXtension Procedure Call (XPC)
  – Copy-in/Copy-out
  – Wrappers

• Recovery
  – Hardware and software checks
  – Stop / Unload and GC / Reload
Some Limitations

• Blame the processor
• Blame the operating system
• Blame us
Outline

• Vision
• Design
• Evaluation
  – Reliability
  – Performance
  – Implementation Cost
• Summary
Tested Drivers

• Sound card drivers
  – SoundBlaster 16 (sb)
  – Ensoniq 1371

• Network drivers
  – Intel Pro/1000 Gigabit Ethernet (e1000)
  – AMD PCnet32 10/100 Mb Ethernet (pcnet32)
  – 3COM 3c90x 10/100 Mb Ethernet
  – 3Com 3c59x 10/100 Mb Ethernet

• Filesystems
  – VFAT Windows-compatible filesystem (vfat)

• Other
  – kHTTPd in-kernel web server (khttpd)
Reliability Test Methodology

1. Load driver
2. Inject bugs
3. Test
   - Nothing
   - Failure
   - Reboot
Reliability Test Methodology

Load driver

Inject bugs

Test

- Nothing
- Failure
- Recovery
- Reboot
Nooks Stops Crashes

Number of crashes

Extension

pcnet32

119

No Nooks

Nooks
Nooks Stops Crashes

Number of crashes

pcnet32

Extension

No Nooks
Nooks

119
0
Nooks Stops Crashes

Number of crashes

Extension

pcnet32  e1000

No Nooks

Nooks

119  52

0
Nooks Stops Crashes

![Bar chart showing the number of crashes for different extensions with and without Nooks.](chart.png)
Nooks Stops Crashes

Number of crashes

<table>
<thead>
<tr>
<th>Extension</th>
<th>No Nooks Count</th>
<th>Nooks Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>pcnet32</td>
<td>119</td>
<td>0</td>
</tr>
<tr>
<td>e1000</td>
<td>52</td>
<td>0</td>
</tr>
<tr>
<td>sb</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

- pcnet32: 119 crashes without Nooks, 0 crashes with Nooks
- e1000: 52 crashes without Nooks, 0 crashes with Nooks
- sb: 10 crashes without Nooks, 1 crash with Nooks
Performance

- Dominant cost is XPC
  - Performance depends frequency of interaction with kernel
Relative Performance

Perf. Relative to Native Linux

150 XPC/sec

Workload

Play MP3
Receive
Stream
Send
Stream
Apache
SpecWeb
Compile
Local
Simple
Web
Relative Performance

<table>
<thead>
<tr>
<th>Workload</th>
<th>Perf. Relative to Native Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Play MP3</td>
<td>150</td>
</tr>
<tr>
<td>Receive Stream</td>
<td>8,923</td>
</tr>
<tr>
<td>Send Stream</td>
<td>60,352 XPC/sec</td>
</tr>
</tbody>
</table>

Perf. Relative to Native Linux
Relative Performance

Workload

- Play MP3
- Receive Stream
- Send Stream
- Apache SpecWeb
- Compile Local
- Simple Web

Relative Performance

- sb
- e1000
- e1000

XPC/sec

- 150
- 8,923
- 60,352
- 1,960
 Relative Performance

<table>
<thead>
<tr>
<th>Workload</th>
<th>Perf. Relative to Native Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Play MP3</td>
<td>150</td>
</tr>
<tr>
<td>Receive Stream</td>
<td>8,923</td>
</tr>
<tr>
<td>Send Stream</td>
<td>60,352</td>
</tr>
<tr>
<td>Apace SpecWeb</td>
<td>1,960</td>
</tr>
<tr>
<td>Compile Local</td>
<td>22,653</td>
</tr>
<tr>
<td>Simple Web</td>
<td>61,183</td>
</tr>
</tbody>
</table>

XPC/sec
Implementation Cost

• Changes to old code
  – Kernel: 924 out of 1.1 million lines
  – Device drivers+VFAT: 0 out of 33,000 lines
  – kHTTPd: 13 out of 2,000 lines

• New code
  – Nooks reliability layer: 22,266 lines
Summary

• Nooks provides a new reliability layer between drivers and the OS
• Nooks prevents 99% of tested faults that cause Linux to crash
• Nooks imposes a modest performance cost
Questions?

Thanks to
Doug Buxton, Steve Martin, Christophe Augier
Microsoft

www.cs.washington.edu/homes/mikesw/nooks
Why didn’t we use a microkernel?

• Doesn’t address our limitations
  – Isolation not much better
  – Fault detection not much better
  – Recovery not much better
  – Doesn’t improve performance

• Requires more changes to the kernel

• Makes compatibility more difficult
Nooks Catches Bugs

Number of failures

Extension

<table>
<thead>
<tr>
<th>Extension</th>
<th>No Nooks</th>
<th>Nooks</th>
</tr>
</thead>
<tbody>
<tr>
<td>sb</td>
<td>156</td>
<td>56</td>
</tr>
<tr>
<td>e1000</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>pcnet32</td>
<td>33</td>
<td>31</td>
</tr>
<tr>
<td>VFAT</td>
<td>232</td>
<td>80</td>
</tr>
<tr>
<td>kHTTPd</td>
<td>175</td>
<td>27</td>
</tr>
</tbody>
</table>
Future work

• Improve performance
• Better recovery
• Automate wrapper generation