Effectiveness of Driver Isolation and Testing in User Space

TECHNISCHE UNIVERSITÄT DARMSTADT

11th EuroSys Doctoral Workshop (EuroDW 2017), April 23, 2017, Belgrade, Serbia

Oliver Schwahn
os@cs.tu-darmstadt.de

PhD Advisor: Prof. Neeraj Suri

DEEDS Group
Department of Computer Science
TU Darmstadt, Germany

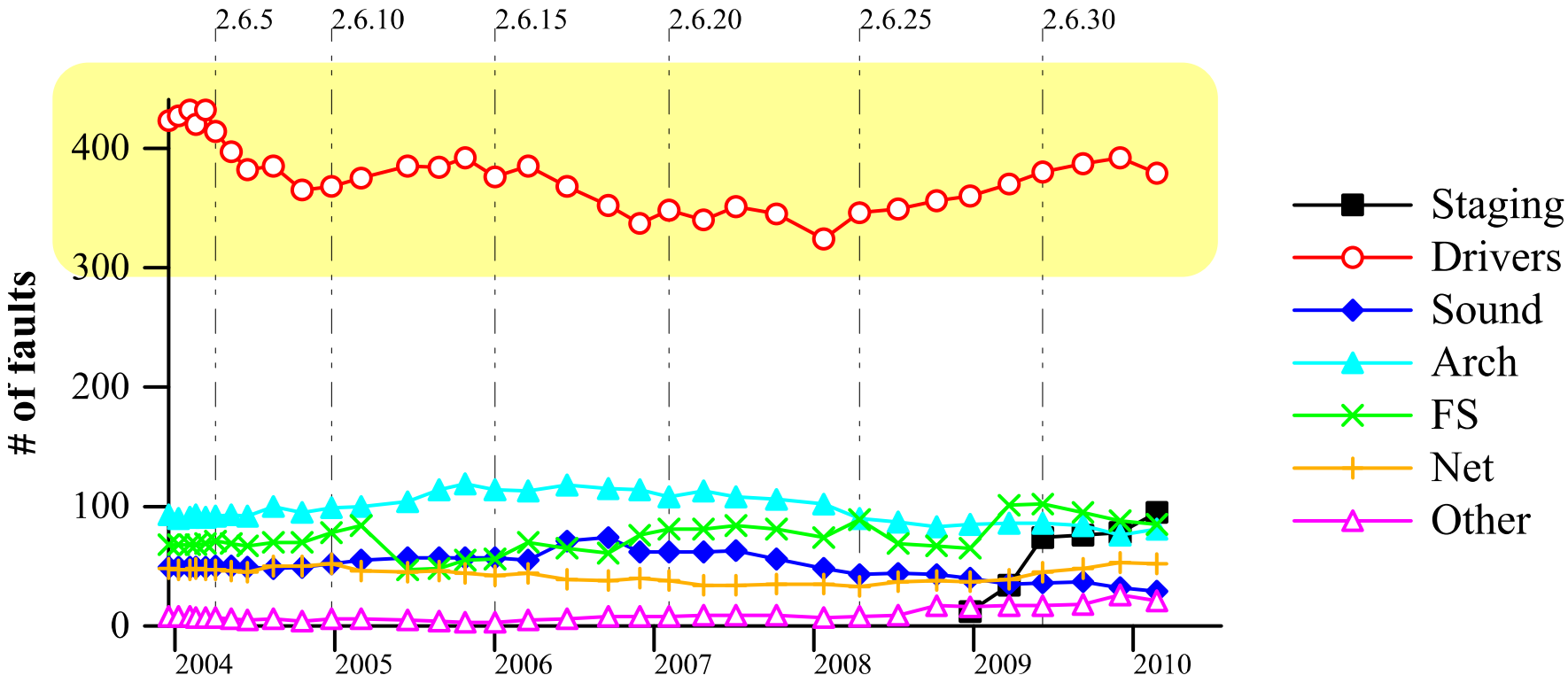
www.deeds.informatik.tu-darmstadt.de

DEEDS Group

```
sgbc_tx raid1 raid0 multipath linear md_mod sd_mod crc_t10dif ata_generic usbhid hid mptsas mptscsih mptbase
uhci_hcd ata_piix libata ehci_hcd scsi_mod usbcore bnx2 nls_base [last unloaded: scsi_wait_scan]
[144516.020026]
[144516.020026] Pid: 0, comm: kworker/0:0 Not tainted 2.6.38-1-amd64 #1 IBM IBM eServer BladeCenter HS21 -[88
e
[144516.020026] RIP: 0010:[<ffffffff812971de>]  [<ffffffff812971de>] ip_options_compile+0x1c0/0x41a
[144516.020026] RSP: 0018:ffff8800cfc83bc0  EFLAGS: 00010286
[144516.020026] RAX: 0000000000000008 RBX: ffff8800a5aed100 RCX: ffff8801282e2077
[144516.020026] RDX: 000000000000000b RSI: 0000000000000134 RDI: ffffffff8180b500
[144516.020026] RBP: ffff8801282e2075 R08: ffffffffa041bdbb R09: ffff8800cfc83d18
[144516.020026] R10: ffffffff8104c5a3 R11: dead000000200200 R12: ffff8800a5aed128
[144516.020026] R13: 0000000000000027 R14: ffff8801282e2060 R15: 0000000000000027
[144516.020026] FS:  00000000000000000(0000) GS:ffff8800cfc80000(0000) knlGS:0000000000000000
[144516.020026] CS:  0010 DS: 0000 ES: 0000 CR0: 000000008005003b
[144516.020026] CR2: 0000000000000134 CR3: 00000001284ac000 CR4: 00000000000006e0
[144516.020026] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[144516.020026] DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
[144516.020026] Process kworker/0:0 (pid: 0, threadinfo ffff88012b6d2000, task ffff88012b65ca40)
[144516.020026] Stack:
[144516.020026]  ffff8801800000000 ffff8800cfc83bd0 0000000000000134 ffffffff8180b500
[144516.020026]  ffff880129a4a900 ffffffffa0420dab ffffffff00000001 ffff8800a5aed128
[144516.020026]  ffff8800a5aed100 ffff880128084000 ffff8801282e2060 ffff880128084000
[144516.020026] Call Trace:
[144516.020026]  <IRQ>
[144516.020026]  [<ffffffffa0420dab>] ? br_nf_pre_routing_finish+0x236/0x258 [bridge]
[144516.020026]  [<ffffffffa041ff36>] ? br_parse_ip_options+0x131/0x19c [bridge]
[144516.020026]  [<ffffffffa0420a6b>] ? br_nf_pre_routing+0x410/0x51a [bridge]
[144516.020026]  [<ffffffff8128f304>] ? nf_iterate+0x41/0x7e
[144516.020026]  [<ffffffffa041bdbb>] ? br_handle_frame_finish+0x0/0x1c4 [bridge]
[144516.020026]  [<ffffffff8128f3a3>] ? nf_hook_slow+0x62/0xde
[144516.020026]  [<ffffffffa041bdbb>] ? br_handle_frame_finish+0x0/0x1c4 [bridge]
[144516.020026]  [<ffffffffa041bdbb>] ? br_handle_frame_finish+0x0/0x1c4 [bridge]
[144516.020026]  [<ffffffffa041bda1>] ? T.865+0x3c/0x56 [bridge]
[144516.020026]  [<ffffffffa041c107>] ? br_handle_frame+0x188/0x1a1 [bridge]
[144516.020026]  [<ffffffffa041bf7f>] ? br_handle_frame+0x0/0x1a1 [bridge]
[144516.020026]  [<ffffffff8126bd81>] ? __netif_receive_skb+0x338/0x4d6
[144516.020026]  [<ffffffff8126c00b>] ? process_backlog+0xec/0x1c7
[144516.020026]  [<ffffffff812700a9>] ? net_rx_action+0xa8/0x206
[144516.020026]  [<ffffffff8104c6ef>] ? __do_softirq+0xc3/0x1a0
-
```

Device Drivers
Major Source of Reliability Issues

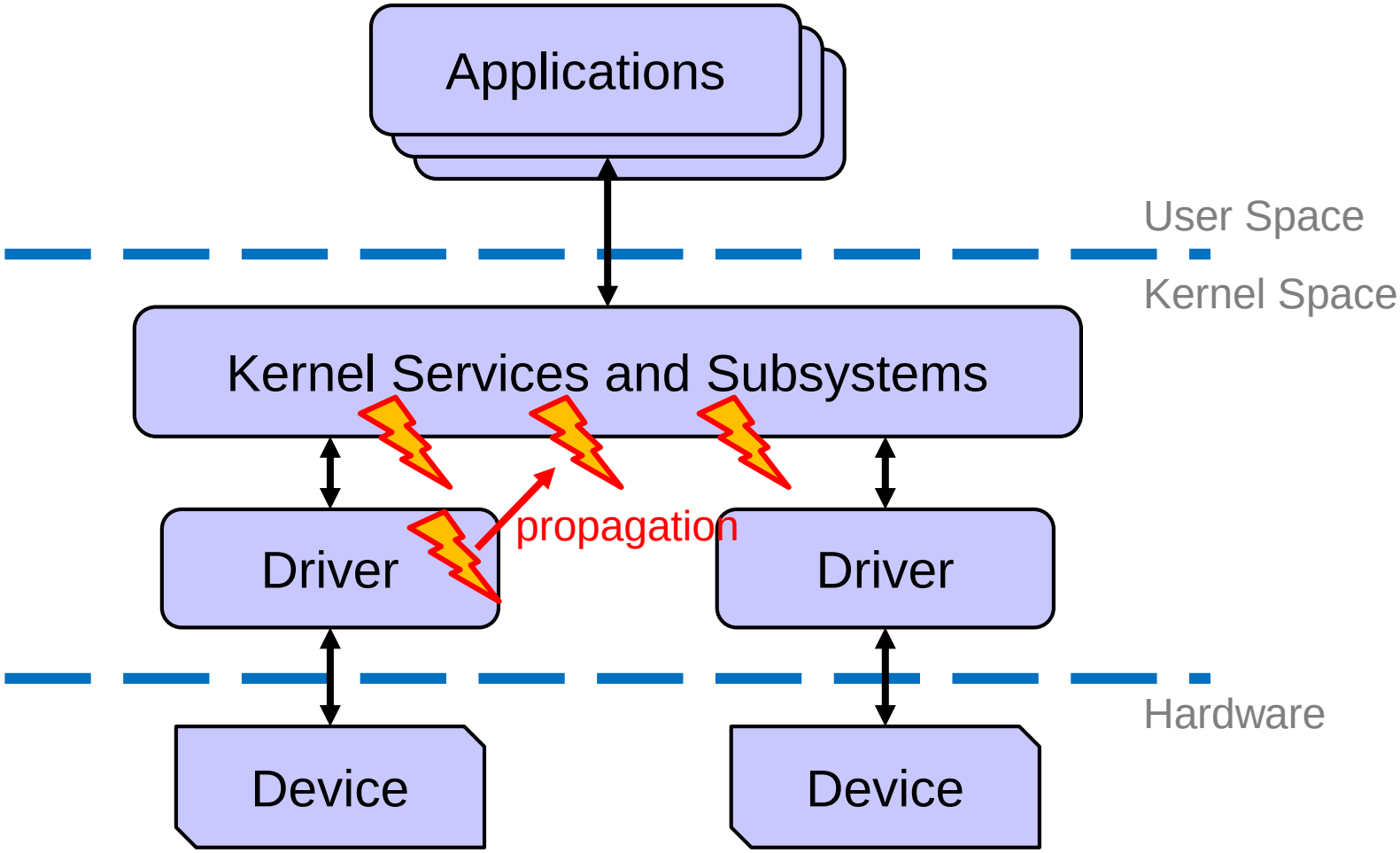
TECHNISCHE UNIVERSITÄT DARMSTADT



Palix et al., “Faults in Linux: Ten Years Later”, ASPLOS 2011

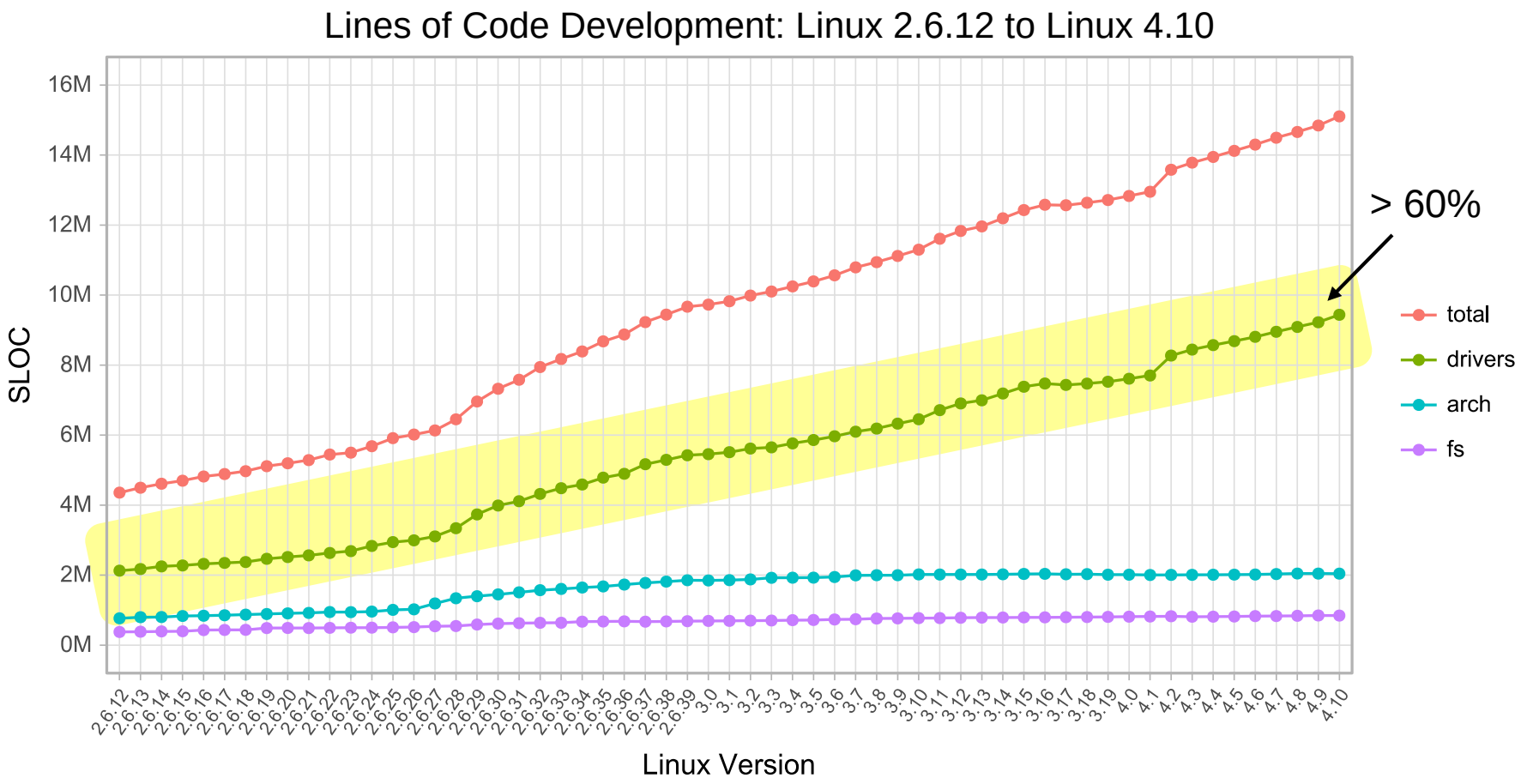
DEEDS Group

Tight Coupling: Drivers & Kernel



Device Drivers
Major Contributor to Growing Complexity

TECHNISCHE UNIVERSITÄT DARMSTADT



generated using 'SLOCCount' by David A. Wheeler

DEEDS Group

Testing in the Kernel Difficult

TECHNISCHE UNIVERSITÄT DARMSTADT

It is **compile tested** only because I did not find an easy way **how to run the code**. Well, it should be pretty safe given the nature of the change.

Note, only **compile tested** this as **do not have any hardware** with it in.

I am touching a lot of arch specific code here and I hope I got it right but as a matter of fact I even **didn't compile test** for some archs as I **do not have cross compiler** for them. Patches should be quite trivial to review for stupid compile mistakes though.

— Excerpts from Linux commit messages

DEEDS Group

Quick Summary

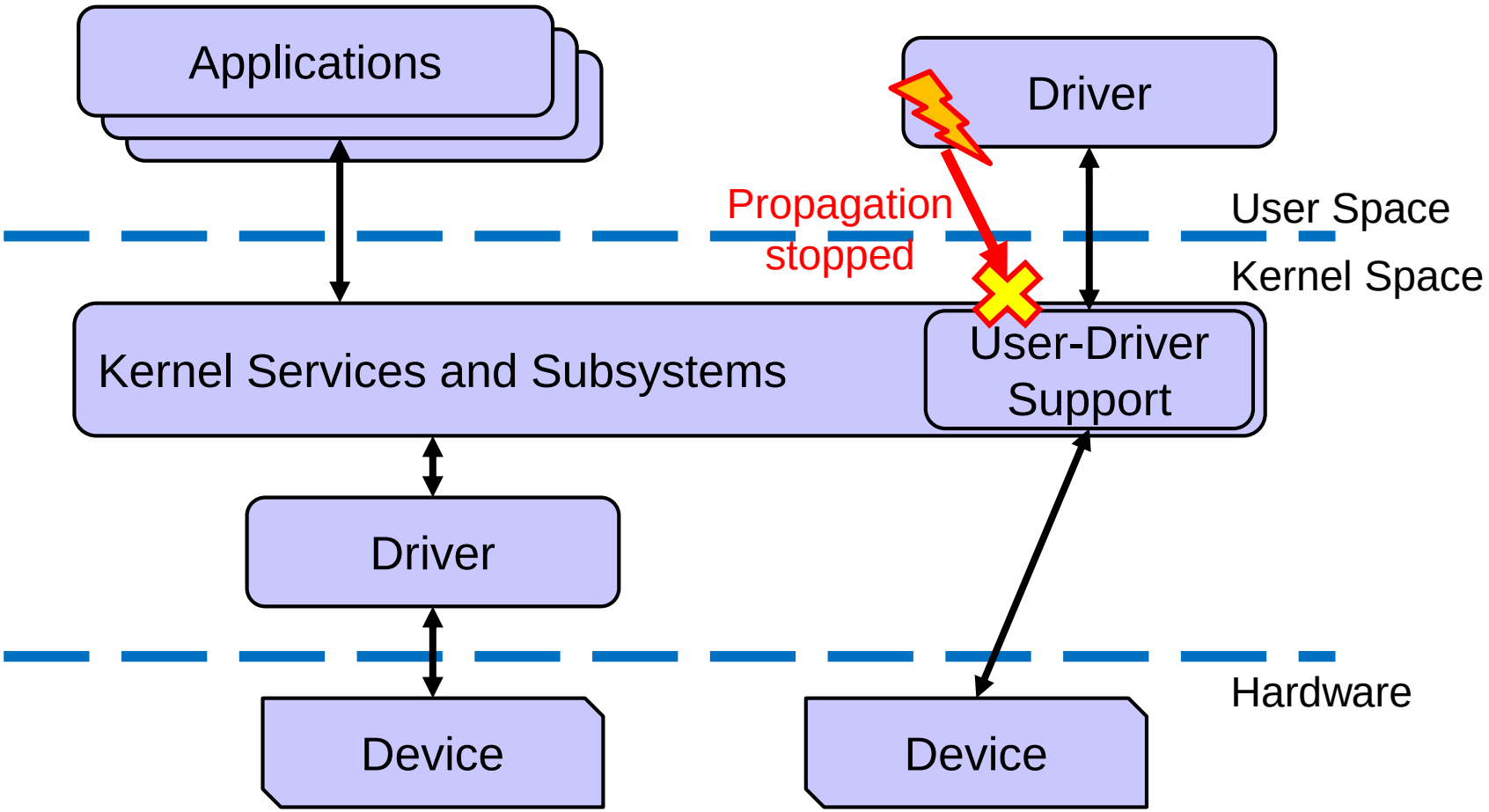
TECHNISCHE UNIVERSITÄT DARMSTADT

- Drivers cause reliability issues
- Increasing complexity
- In-kernel development difficult
- Lack of testing

- Focus on 2 directions
  - Tolerate faulty drivers
  - Improve driver code

DEEDS Group

Tolerating Faulty Drivers: Isolation in User Space



Existing Isolation Approaches

TECHNISCHE UNIVERSITÄT DARMSTADT

- Several approaches exist
  - None of them has been widely adopted

➔ Why is that the case?

  - Are they ineffective
  - Too complex
  - Performance impact too high

DEEDS Group

Are Existing Techniques Ineffective?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Unclear how well effects of different fault types are contained
  - Thorough study missing
  - In practice, we may observe surprising effects
  - Which mechanisms are effective against which faults?
    - Protocol violations, hardware faults, memory safety & concurrency bugs…

➔ Fault injection study

DEEDS Group

Isolation Cost Too High?

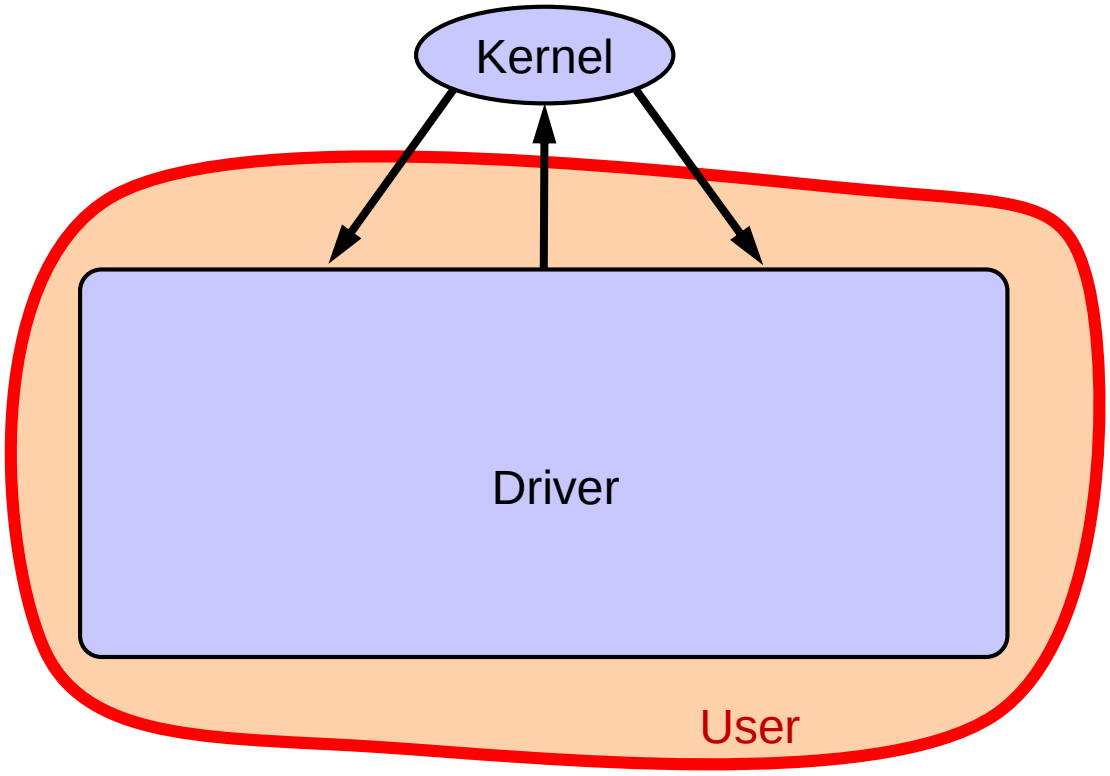
TECHNISCHE UNIVERSITÄT DARMSTADT

- Isolation imposes overhead
  - Additional software layer
- Overhead may vary with different usage scenarios
- Practical trade-off: Isolate less code, gain performance

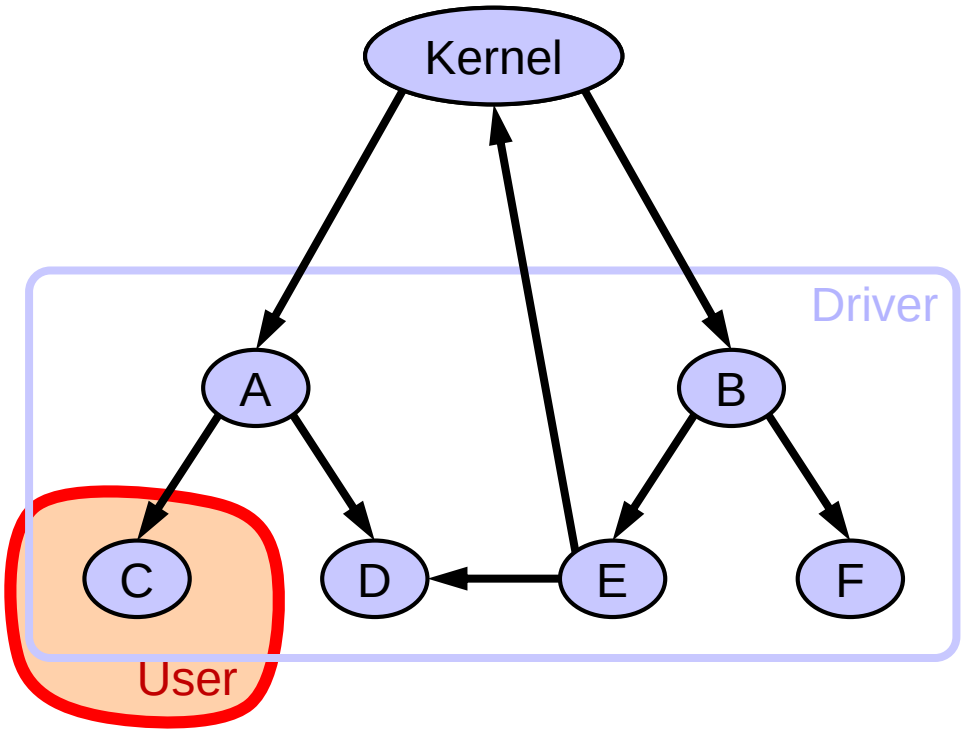
➔ Fine-tune isolation to dynamic usage scenario

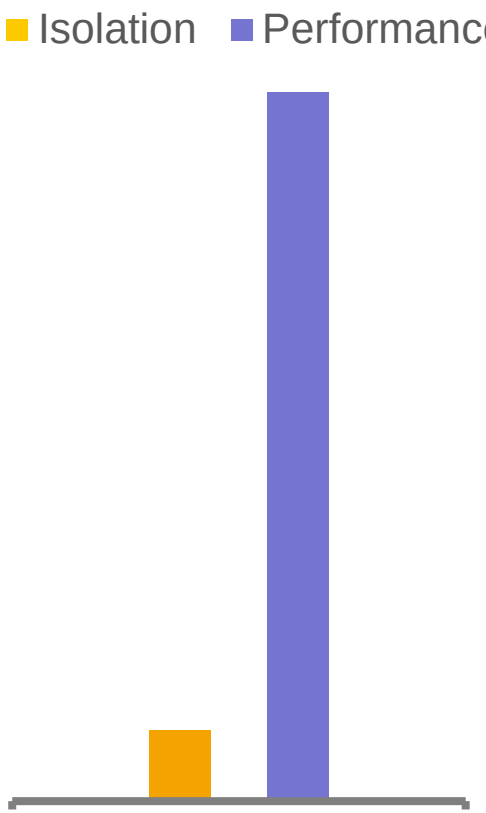
DEEDS Group

Trade-off: Isolation vs Performance

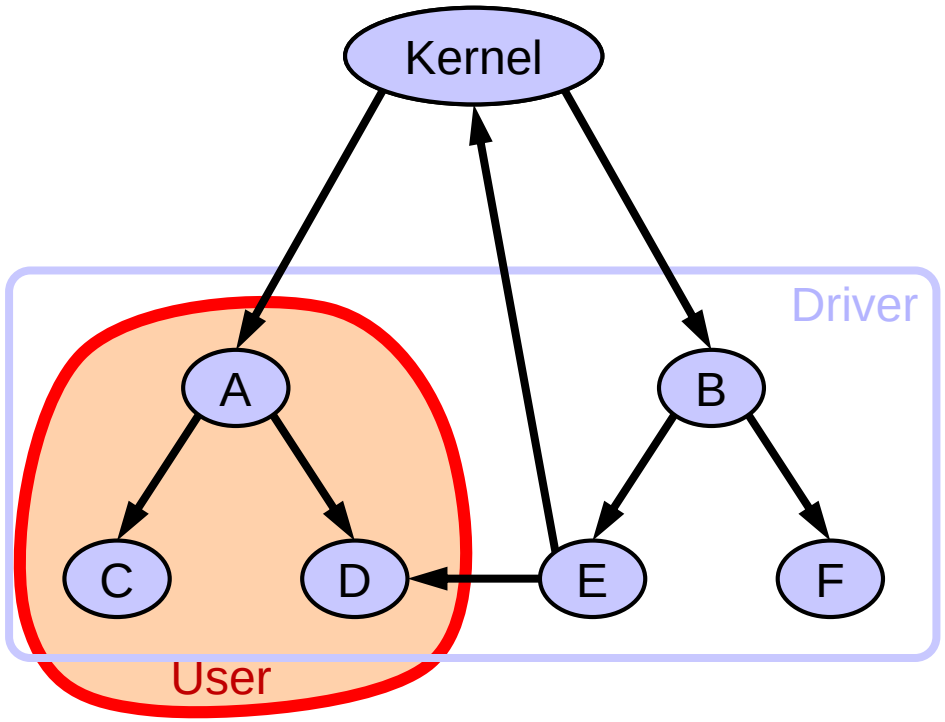


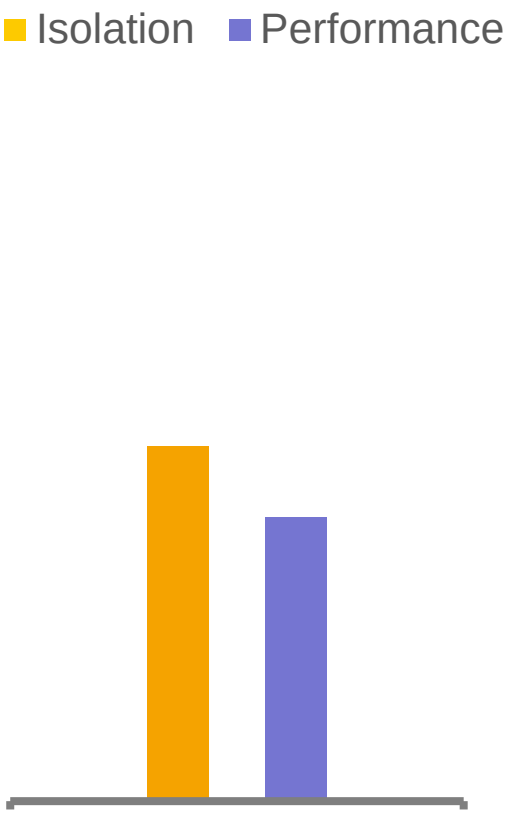
Trade-off: Isolation vs Performance



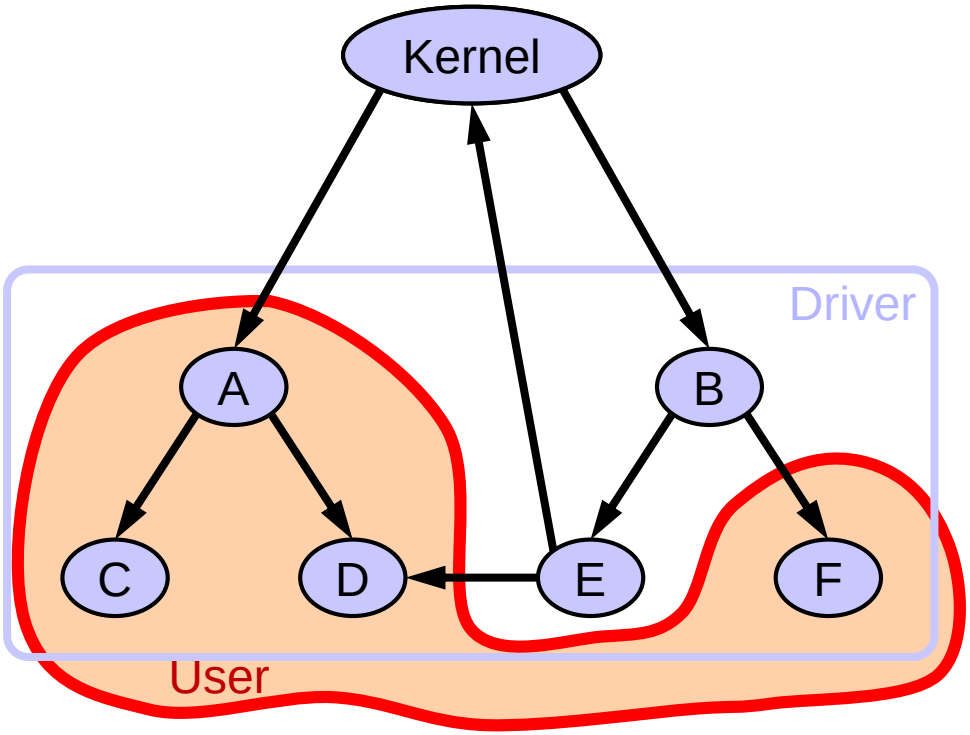


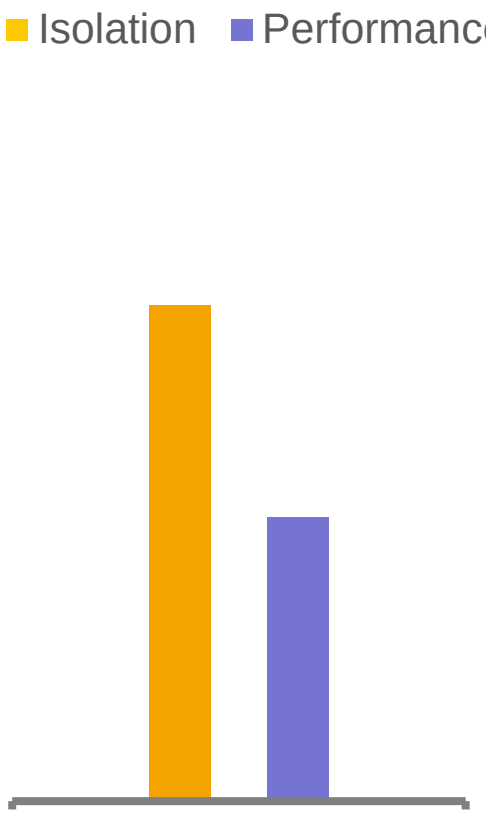
Trade-off: Isolation vs Performance





Trade-off: Isolation vs Performance





Isolation as Ultimate Answer?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Isolation contains faults in the driver
  - Driver crashes still annoying
  - Effects on overall system
    - Down times for recovery or restart
    - Decrease functionality, data loss

➔ Isolation treats the symptoms, not the root cause

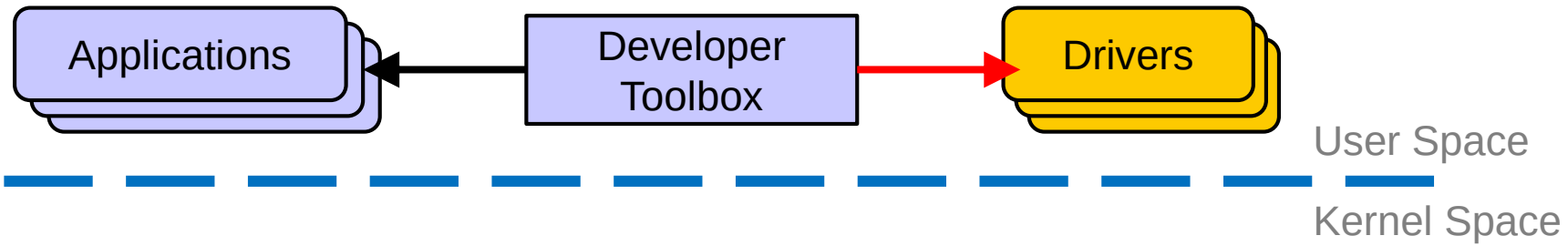
DEEDS Group

Treating the Root Cause

TECHNISCHE UNIVERSITÄT DARMSTADT

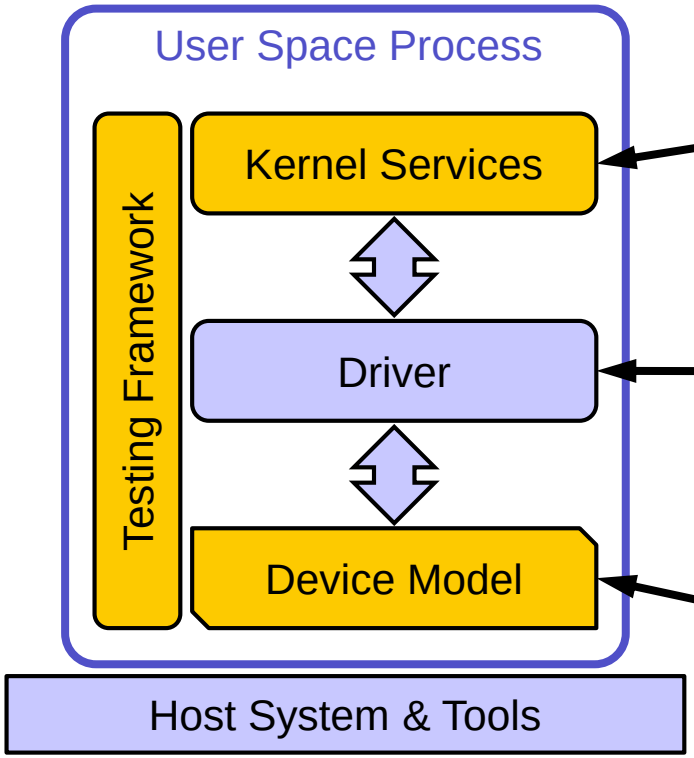
- We have drivers isolated in user space
- We have a large toolbox for user space development
  - Debugging
  - Testing
  - Code analysis

➔ Use the power of the user space toolbox for drivers



DEEDS Group

Testing in User Space



- Simple Mocking
- Learn from tracing
- User-mode Linux

- Driver “as library”
- Test in isolation
- Test individual functions
- Test for code coverage

- Simple mocking
- Learn from tracing
- Infer from specification

Thank You